University of Crete
School of Sciences and Engineering
Computer Science Department

IDENTIFYING TRENDS IN RECOMMENDATION ALGORITHMS

by

IOANNIS ROUSIDIS

Master's Thesis

Heraklion, March 2007

University of Crete
School of Sciences and Engineering
Computer Science Department

IDENTIFYING TRENDS IN RECOMMENDATION ALGORITHMS

by

IOANNIS ROUSIDIS

A thesis submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Author:

_____
Ioannis Rousidis, Computer Science Department

Supervisory
Committee:

_____
Dimitris Plexousakis, Professor, Supervisor

_____
Grigoris Antoniou, Professor, Member

_____
Tzitzikas Ioannis, Assistant Professor, Member

Approved by:

_____
Panos Trahanias, Professor
Chairman of the Graduate Studies Committee

Heraklion, March 2007

Abstract

Digital and Social Networking Revolution plays a major role in our everyday life. The social prevalence of this can be evidenced by the evolution of, and demand for, personalized media and on-line shopping. *Recommender Systems* (RS) are applications that provide personalized advice to users about products or services they might be interested in.

However, the ultimate effectiveness of an RS is dependent on factors concerning the underlying algorithm. In this work we report on the algorithmic aspects of the *collaborative filtering* (CF) method, the prevalent method for providing recommendations. More specifically, we focus on three key factors of the CF: *efficacy*, *efficiency* and *accommodation* to new data. Efficacy involves the quality of the recommendation result. Moreover, especially CF suffers from slow response time, because each single prediction requires the scanning of a whole database of user ratings, making thus efficiency a hot topic. Finally, a RS must be capable of handling new data, be it new users or new items.

To provide high-quality recommendations, we utilize a theoretical framework for combining information from different sources. Our approach differs in that it is purely probabilistic. The two combinations schemes are being examined: the sum and the product rule. We experimented with our schemes using in each single prediction the following sources: user ratings, item ratings and the categories they belong. Our experimental evaluation indicates that our framework improves the quality performance of the CF method.

In order to improve the efficiency and accommodate to new data, we propose the *Incremental Trend Diagnosis* (ITD), a novel framework for CF. This approach uses dimensionality reduction to create a model on user rating trends which is updated incrementally. CF is applied upon trends for the estimation of user-to-user similarities. Thus, we replace complex similarity estimations with a scalar operation upon rating trends. Experimental results show that our framework is capable of formulating high quality recommendations which makes ITD suitable for online application. To our

knowledge, little work has addressed the use of incremental updating model in collaborative filtering.

Supervisor: Dimitris Plexousakis
Professor

Περίληψη

Η επανάσταση στα ψηφιακά και κοινωνικά δίκτυα παίζει σημαντικό ρόλο στην καθημερινή μας ζωή. Η εξάπλωσή τους στην κοινωνία μπορεί να τεκμηριωθεί από την εξέλιξη των εξατομικευμένων μέσων ενημέρωσης και ψυχαγωγίας και των ηλεκτρονικών αγορών. Τα *Συστήματα Συστάσεων* (ΣΣ) είναι εφαρμογές που παρέχουν εξατομικευμένες συμβουλές στους χρήστες για προϊόντα ή υπηρεσίες που πιθανόν να τους ενδιαφέρουν.

Παρολαυτά, η τελική αποτελεσματικότητα ενός ΣΣ εξαρτάται από παράγοντες που σχετίζονται με τον υποκείμενο αλγόριθμο. Σε αυτήν την εργασία εξετάζουμε τις αλγοριθμικές πτυχές της *Συνεργατικής Διήθησης* (ΣΔ), της επικρατέστερης μεθόδου για την παροχή συστάσεων. Πιο συγκεκριμένα εστιάζουμε σε τρεις παράγοντες κλειδιά: *αποτελεσματικότητα, αποδοτικότητα* και *προσαρμογή* σε νέα δεδομένα. Η αποδοτικότητα αφορά την ποιότητα στις συστάσεις. Επίσης, ειδικά η ΣΔ πάσχει από αργό χρόνο απόκρισης λόγω της απαίτησης για έλεγχο όλης της βάσης δεδομένων σε κάθε σύσταση, καθιστώντας την αποδοτικότητα φλέγον θέμα. Τέλος, ένα ΣΣ πρέπει να είναι σε θέση να μπορεί να προσαρμόζεται σε νέα δεδομένα, είτε πρόκειται για νέους χρήστες είτε για νέα αντικείμενα.

Για την παροχή συστάσεων υψηλής ποιότητας, αξιοποιήσαμε ένα θεωρητικό πλαίσιο εργασίας για συνδυασμό πληροφορίας από διαφορετικές πηγές. Η προσέγγιση μας διαφέρει από τις ήδη υπάρχουσες στο ότι είναι αμιγώς πιθανοκρατική. Τα δύο συνδυαστικά σχήματα που εξετάζονται είναι: ο κανόνας του αθροίσματος και του γινομένου. Εξετάσαμε πειραματικά τα σχήματά μας χρησιμοποιώντας για κάθε πρόβλεψη τις ακόλουθες πηγές: τις βαθμολογίες του χρήστη, του αντικειμένου και τις κατηγορίες που αυτό ανήκει. Η πειραματική αποτίμηση υποδεικνύει ότι το θεωρητικό πλαίσιο το οποίο προτείνουμε βελτιώνει σημαντικά την ποιοτική απόδοση της ΣΔ.

Με σκοπό τη βελτίωση της αποτελεσματικότητας και την προσαρμογή σε νέα δεδομένα, προτείνουμε την *Αυξητική Διάγνωση Τάσεων* (ΑΔΤ), ένα νέο πλαίσιο εργασίας για τη ΣΔ. Αυτή η προσέγγιση χρησιμοποιεί μείωση διαστάσεων για τη δημιουργία ενός μοντέλου πάνω στις τάσεις βαθμολογίας των χρηστών το οποίο

ενημερώνεται αυξητικά. Η ΣΔ εφαρμόζεται πάνω στις τάσεις για τον υπολογισμό ομοιότητας μεταξύ των χρηστών. Με αυτό τον τρόπο αντικαθιστούμε τις πολύπλοκους υπολογισμούς ομοιότητας με μια μονόμετρη πράξη. Πειραματικά αποτελέσματα δείχνουν ότι το πλαίσιο εργασίας μας μπορεί να προσφέρει συστάσεις υψηλής ποιότητας καθιστώντας την ΑΔΤ κατάλληλη για ηλεκτρονική εφαρμογή. Προς γνώση μας, ελάχιστες εργασίες έχουν κάνει χρήση, ως τώρα, αυξητικά ενημερωμένου μοντέλου στη ΣΔ.

Επόπτης: Δημήτρης Πλεξουσάκης
Καθηγητής

Στους γονείς μου, Μαρία και Βασίλη, τον αδερφό μου Πασχάλη για την αγάπη και την υποστήριξη τους σε κάθε πτυχή της ζωής μου.

Ευχαριστίες

Ιδιαίτερες ευχαριστίες αξίζουν στον επόπτη μου κ. Δημήτρη Πλεξουσάκη, για όσα μου προσέφερε αυτά τα τρία χρόνια της συνεργασίας μας και για τις ευκαιρίες που μου έδωσε. Χωρίς την ουσιαστική του καθοδήγηση και τις επισημάνσεις του η ολοκλήρωση αυτής της εργασίας θα ήταν αδύνατη.

Ακόμα θα ήθελα να ευχαριστήσω τον κ. Γιάννη Τζίτζικα καθώς και τον κ. Γρηγόρη Αντωνίου για την προθυμία τους να συμμετάσχουν στην επιτροπή για την αξιολόγηση της εργασίας αυτής καθώς και για τις επισημάνσεις τους πάνω σε αυτή. Ένα επίσης ευχαριστώ θα ήθελα να απευθύνω στην κ. Ρένα Καλαϊτζάκη, γραμματέα μεταπτυχιακών σπουδών, για την ευγένεια και κυρίως την υπομονή της απέναντι μου όλα αυτά τα χρόνια.

Η πρώτη μου επαφή με την περιοχή των συστημάτων συστάσεων έγινε μέσα από μία συνεργασία με το Μάνο Παπαγγελή από την οποία και προέκυψε η πτυχιακή μου εργασία. Για αυτή, λοιπόν, τη "μύηση" θα ήθελα Μάνο να εκφράσω την ευγνωμοσύνη μου, όπως επίσης και στο πρόσωπο του Γιώργου Τζαγκαράκη του οποίου οι υποδείξεις, παρατηρήσεις, συμβουλές αποδείχτηκαν πολύτιμες για τη διεκπεραίωση αυτής της εργασίας ενώ μέσα από τις συζητήσεις μαζί του γεννήθηκαν όμορφες ιδέες. Γιώργο δουλεύοντας μαζί σου νομίζω ότι κέρδισα πολλά πράγματα και γι' αυτό το λόγο σε ευχαριστώ ολόψυχα!

Ένα μεγάλο ευχαριστώ ανήκει σε όλους τους συμφοιτητές και συναδέλφους με τους οποίους συνεργάστηκα καθ' όλη την διάρκεια των σπουδών μου. Αισθάνομαι τυχερός που μερικές από τις συνεργασίες κατέληξαν σε πραγματικές φιλίες. Ευχαριστώ λοιπόν όλους όσους στάθηκαν πλάι μου όλα αυτά τα χρόνια για τις εμπειρίες που μοιραστήκαμε και θα θυμόμαστε για όλη μας τη ζωή. Ιδιαίτερα θέλω να ευχαριστήσω τους Ρούσση Δαδαμόγια και Τάσο Βλαϊκίδη, για τη στενή φιλία τους όλα αυτά τα χρόνια ακόμα και τώρα που έχουν φύγει πλέον από το Ηράκλειο, και, τέλος, τους Ζωή Πολιτοπούλου, Νίκο Παπαχριστοδούλου, Γιώργο Κωνσταντινίδη, Παναγιώτη Παπαδάκο και Γιάννη Θεοχάρη παρέα με τους οποίους τα πολλά ατέλειωτα ξενύχτια στα υπόγεια εργαστήρια μόνο βαρετά δεν ήταν!

Όμως το τελευταίο αλλά και μεγαλύτερο ευχαριστώ ανήκει στην οικογένειά μου και πιο συγκεκριμένα στους γονείς μου Βασίλη και Μαρία καθώς και στον αδερφό μου Πασχάλη που ήταν πάντα δίπλα μου και με στήριξαν σε όλες τις δυσκολίες. Η δύναμη που μου έδωσε η συνεχής υποστήριξή τους όλα αυτά τα χρόνια δεν μπορεί να περιγραφεί σε λίγες γραμμές. Για το λόγο αυτή η εργασία αυτή είναι αφιερωμένη σ' αυτούς και ελπίζω να αποτελέσει μια μικρή ανταμοιβή για τις θυσίες και τις προσπάθειες που έχουν κάνει όλον αυτόν τον καιρό.

Με την παράδοση αυτής της μεταπτυχιακής εργασίας μια πορεία περίπου επτά χρόνων φτάνει στο τέλος της, τουλάχιστον σαν φοιτητής, στο Πανεπιστήμιο Κρήτης. Ένα ταξίδι που δε νομίζω ότι ξεκίνησε με τους καλύτερους οιωνούς. Παρόλαυτα όμως μου χάρισε αναμνήσεις και μου έμαθε πράματα τα οποία θα κρατήσω για μια ζωή όπως υπομονή, επιμονή, θετική και αισιόδοξη στάση απέναντι σε όλα.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Digital and Social Networking Revolution plays a major role in our everyday life. More and more people dive into the long tail of content discovery struggling to manage the content overload. The social prevalence of this can be evidenced by the evolution of, and demand for, personalized radio, television, video and on-line shopping. Conflicting needs and differences in personal preferences, social and educational backgrounds, stress the need for personalized intelligent systems that process, filter, and display available information in a manner that suits each individual using them.

Recommender Systems (RS) are applications that provide personalized advice to users about products or services they might be interested in. From a user's perspective, an effective recommender system inspires trust in the system, has system logic that is at least somewhat transparent, points users towards new, not-yet-experienced items, provides details about recommended items, including pictures and community ratings, and finally, provides ways to refine recommendations by including or excluding particular genres. Users will probably express willingness to provide more input to the system in return for more effective recommendations.

However, the ultimate effectiveness of an RS is dependent on factors concerning the underlying algorithm. In this work we report on the algorithmic aspects of recommendation technologies which have been a popular topic of research ever since the ubiquity of the web made it clear that people of hugely varying backgrounds would be able to access and query the same underlying data. The initial human computer interaction challenge has been made even more challenging by the

observation that customized services require sophisticated data structures and well thought-out architectures to be able to scale up to thousands of users and beyond.

More specifically, we focus on three key factors of the recommender technologies: *efficacy*, *efficiency* and *accommodation* to new data. Efficacy involves the quality of the recommendation result and has gained much attention since various methods have been proposed for improvement. Most of the recommendation algorithms suffer from slow response time, because each single prediction requires the scanning of a whole database of user ratings, making thus efficiency a hot topic. Finally, a RS must be capable of handling new data, be it new users or new items.

1.2 Contributions

The primary contributions of this thesis are:

- A formal notation of the recommendation framework and a literature about the current state-of-the-art technologies that have emerged
- The development of a common theoretical framework for combining recommendation sources which are used jointly to make a prediction improving recommendation quality. Also an experimental comparison of the two basic combination schemes, the sum and the product rule in sparse environment
- The development and assessment of a collaborative filtering based methodology to improve the efficiency of recommendation algorithms in online applications
- The development and assessment of a methodology for incrementally updating to new data and dealing with the computational cost required by batch-mode algorithms often referred as the scalability problem
- A roadmap for future extensions in recommendation technologies

1.3 Structure

In this thesis, we study the algorithmic foundations of recommendation technologies and provide methods to address specific shortcomings that harmfully affect their further adoption into commerce and research applications. The remainder of the thesis is organized as follows:

Chapter 2 introduces recommendation algorithms, formulates the recommendation problem, classifies the existing recommendation approaches and presents specific shortcomings that need to be addressed. Some of these limitations, we are going to deal with, are also referred as sparsity, cold-start and scalability. Moreover this chapter presents and discusses the work done so far towards overcoming these limitations.

Chapter 3 stresses the need to extend the classic CF approach. To provide high-quality recommendations, we utilize a theoretical framework for combining information from different sources. Our approach differs in that it is purely probabilistic. The two combinations schemes are being examined: the sum and the product rule. We experimented with our schemes using in each single prediction the following sources: user ratings, item ratings and the categories they belong. Our experimental evaluation indicates that our framework provides better quality results in recommendations compared to naïve linear combinations. However, it's worth noticing that in most cases sum-rule, in its optimum configurations, has slightly outperformed product-rule. The main reason is the sensitivity in errors which is intense in the latter case due to factorization of prediction techniques.

Chapter 4 presents a new framework for collaborative filtering migrating from user ratings to user trends. Information about user trends based on rating activity can be distilled by using an incremental PCA algorithm to map users into a dynamic traceable subspace of the item space. The result is the creation of a "thinner" than the original user-rating matrix for further computation of user similarities. This incremental background modeling technique updates user rating trends and replaces expensive vector operations with a scalar operation. Thus, we achieve to speed-up computations of high dimensional user-item matrices with no trade off between recommendation accuracy and response time.

Chapter 5 concludes the research contributions of the thesis, discusses ways to extend the capabilities of recommendation algorithms and draws directions for further research work.

Finally, there is a chapter committed to relative bibliography that is referred throughout this thesis.

Chapter 2

Recommender Systems

2.1 Introduction

Whenever someone wants to use a search engine, make a purchase at an online store or read his own e-mails, he has to confront the information overload problems caused by the rise of internet. Moreover, the online retailers need to keep their customers coming back but also to cross-sell profitable products, so they are compelled to provide a good service. One way to overcome these problems is using recommender systems.

Definition 1: *Recommender system (RS) is a system that guides a user to objects of his interest among a set of selectable items using a knowledge-base which can be learned by other users' preferences or hand coded by experts.*

This definition denotes that the notion of recommendation carries the idea that a specific object has been recommended by another user or an expert. Predictions are made only for items the active user hasn't interacted with previously.

A RS works typically as follows: (1) at first, the recommender system collects any given *information* at one place building *user's profile* and (2) thereafter it applies a *prediction technique*. Predictions can be made either by applying a model learning technique in the dataset (model-based algorithms), e.g. using a *clustering algorithm* [14], [102], or on the fly (memory-based algorithms), e.g. using a *k nearest neighbor algorithm* [14], [79]. A typical prediction could be a requested prediction for an individual object, i.e. a numerical value which represents the amount of user's expected interest for the item, or a list of the *top N* recommendations.

In order to provide information about the state of the prediction technique or its capability in predicting user's interest most RS embody so-called *validity indicators*, i.e. features analogous to using reliability indicators in [10], [100]. Due to variation in prediction techniques, most of them have unique validity indicators. For example, social filtering [39], [92] reveals the number of similar users rated the information, case-based reasoning [47] returns the number of similar rated items by the user.

The elements above describe the basic I/O's of a typical RS it's shown in Figure 2-1.

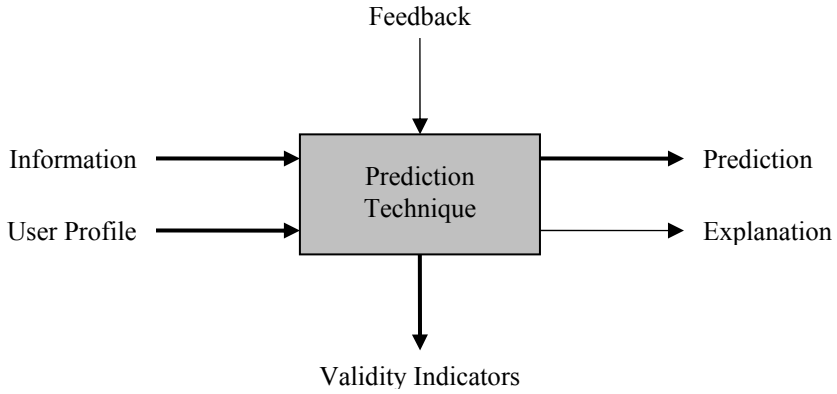


Figure 2-1: A Typical Model of Recommender System

As shown in Figure 2-1, basic I/O's are illustrated with bold arrows while other additional elements also incorporated in this model are indicated by subtle arrows. These elements are described below.

On account of their learning capability, several prediction techniques like [39], [92], [103] can also learn from *feedback* provided by users, which may be either explicit or implicit, in order to provide more accurate prediction in the future. *Explanation data* can be provided optionally by some of the prediction techniques. Explanations guarantee transparency, exposing the data and the reasoning behind a prediction, increasing so the acceptance of prediction systems [39]. Users are now able to understand the reasoning of the prediction and decide whether to accept the outcome or not in case of being inaccurate.

2.2 Collecting Preference Data

Before proceeding in formulation of the prediction problem it is necessary to describe the tactic the RS collect preference data from users. So, in order to tailor personalized recommendations for the active user's needs, systems must collect personal information, for instance user's history of purchase, demographic information, click-stream data, and so on. Given a user u and an item i, his preference to the specific item is expressed traditionally via a rating $r_{u,i}$. Ratings are distinguished in two different categories:

2.2.1 Explicit ratings

According to this category users specify explicitly their preference for any particular object by indicating their ratings. Ratings range from 1, expressing greatest distaste, to a highest point, expressing maximum liking to the object. Then the user's model can be constructed by employing these ratings logged by the system. The more ratings one user provides, the more accurate the predictions/recommendations provided will be. An example of a user-item matrix with explicit ratings is presented in Table 2-1, where ratings scale from 1 to 5. The symbol "$\varnothing$" is used in the table to denote that users did not give a rating to the specific movies.

	i_1	i_2	i_3	i_4
u_1	2	3	4	3
u_2	$\varnothing$	2	3	$\varnothing$
u_3	1	4	$\varnothing$	1
u_4	$\varnothing$	2	3	2

Table 2-1: An example of matrix with explicit ratings (user-item matrix)

2.2.2 Implicit ratings

In this category [52], [64], ratings identify user's preference to specific categories. Users are never actually prompted to express their degree of preference to certain categories justifying so the reason term "implicit" is introduced for.

- **Via explicit rating:** Relying on the fact that an object belongs to a number of different categories a user model can be developed based on these category preferences. Each time a user gives an explicit rating that is considered to be "good" to a specific object then user's model is updated so as to include the preference to the category the object belongs to and vice versa. For this reason a threshold is being used to discriminate a "good" rating from a "bad" one depending on whether the rating is greater than or equal to the threshold or not. An example of a user-item matrix with implicit ratings is presented in Table 2-2, where for each category there are two columns, one including the sum of the "positive" ratings, i.e. ratings greater than or equal to the threshold, and one including the sum of the "negative" ratings, i.e. ratings less than the given threshold.

	C_{1pos}	C_{1neg}	C_{2pos}	C_{2neg}
u_1	36	16	28	13
u_2	0	0	15	7
u_3	22	43	0	0
u_4	18	19	0	0

Table 2-2: An example of matrix with implicit ratings (user-category matrix)

- **Via observation:** Alternative typical examples for implicit ratings are data purchasing, time to read news on Usenet [79] and behaviour browsing [31], [60].

2.2.3 Discussion

Explicit ratings ask for additional efforts on users' behalf leading them to avoid the encumbrance of explicitly stating their preferences. This results in either leaving the system or relying upon "*free-riding*" [3]. Alternatively, gathering preference information by merely observing user's behavior is much less obtrusive [64].

Implicit ratings are easy to collect but imply adding noise to the information collected. For instance, some of the purchases that are gifts do not reflect the active user's interests. For example, "I am a fan of pop bands whereas my friend, Elias, likes metal music. I have no credit card in my possession and just because Elias owns an account in *Amazon.com* I ask him to make orders on my behalf as a favour." Moreover, the inference that purchasing implies liking does not always hold. For example: "The other day I bought a CD of *George Michael* but I didn't like it at all. Is this product going to be recorded in my purchase history anyway? And if yes, will I have to receive recommendations for this type of products in the future?" The difficulty in collecting explicit ratings has made some providers of product recommendation services adopt bilateral approaches. For instance, Amazon.com computes explicit-ratings based recommendations whenever possible. In case of unavailability, implicit ratings from observations are used instead.

2.3 Formal Notation of the Recommendation Framework

The recommendation problem, in its most common formulation, is reduced to the problem of estimating ratings for the unseen items by a user. Speaking in terms of submitted ratings to items or other profile information, prior transactions of the user with the system are the ingredients forming the basis of this estimation.

More formally, the recommendation problem can be formulated as follows. Denote the set of all m users as $U = \{u_x : x = 1, 2, ..., m\}$ and the set of all possible n items that can be recommended, such as books, movies, books or CD's, as $I = \{i_x : x = 1, 2, ..., n\}$.

The space I of possible items can be very large, ranging in hundreds of thousands or even millions of items in some applications, such as recommending books or CDs. Similarly, the user space can also be very large – millions in some cases. Let f the

utility function that measures usefulness of item i to user u, i.e. $f : U \times I \rightarrow R$. As described in previous section, the utility of an item in RS is usually represented by a rating. So, function f maps the two-dimensional space $U \times I$ to the set of real numbers wherever possible.

Definition 2: *A recommendation algorithm is a function that takes as input data from users such as transactions or any other activity indicating personal preferences and calculates predictions for the unrated items by the active user.*

In general utility can be an arbitrary function. Each element in the user space U can be defined by a profile including various user characteristics, such as age, gender, income, marital status, etc. In the simplest case, the profile can contain only a single identifier element, such as *User ID*. Similarly, for each element in the item space I a set of characteristics is defined. For instance, in a recommendation application, where I is a collection of movies, each movie can be represented by its ID, but also by its title, genre, leading actors, year of release, director, etc as well.

The central problem of the RS as described in this section is the fact that function f is usually not defined on the whole $U \times I$ space, but only on a subset of it that consists of existing rating. The prediction function is defined only over the items previously rated by at least one user.

2.4 Types of Recommendation Techniques

Recommendation techniques can be classified in many possible ways [80], [88], [98]. Specifically, considering that recommender systems have (i) background data - information the system has before the beginning of the recommendation process, (ii) input data – information a user must provide to the system in order to receive a recommendation, and (iii) an algorithm combining background and input data to arrive at recommendation results, we can distinguish six different recommendation techniques.

2.4.1 Content Based Techniques

Content-based filtering is deeply rooted in information retrieval [4], [83] and information filtering [9] research. This approach uses the content of the recommended item as a basis for the recommendations and employs many techniques extensively studied in the past.

In order to improve the traditional information retrieval approaches user profiles containing personal information about user needs, preferences and tastes were used. This kind of information can be obtained explicitly, e.g. using questionnaires, or implicitly, by monitoring their transactional behaviour as described in previous section.

Content-based algorithms are mainly used for recommending web pages (URLs), news, jokes, publications or other kinds of text documents. First, information about user preferences is maintained either during the registration process where user gives initial input about his preferences or over the document rating procedure. Then recommendations can be formed by filtering in the documents that better match the user's logged profile or preferences according to their content. The words in an article, [66] for instance, may constitute the content attributes when recommending articles. In another example of a movie recommendation application, the content-based recommendation system in order to recommend movies to a user a system needs to examine user's rating history, i.e. the movies highly rated movies in the past (specific genres, actors, subject matter, directors etc.), to discover movies rather common to his taste. Only movies with high degree of accordance to any of user's preferences are to be recommended.

Moreover, Bayesian classifiers [61], [72] and various machine learning methods, such as decision trees, clustering and artificial neural networks [72] are techniques that have also been used for content based recommendations besides the traditional heuristics based mostly on information retrieval methods. The difference from the retrieval-based methods lies in that calculations to estimate ratings are based not on a heuristic formula, such as the cosine similarity measure, but rather on learning a *model* from the underlying data by using statistical or machine learning. For example, based on a set of Web pages rated by the user as "relevant" or "irrelevant", in [72] they use a naïve Bayesian classifier [29] to classify unrated Web pages.

2.4.2 Collaborative Filtering

Many collaborative systems have been developed in the industry and the academia. The Tapestry system relied on each user to identify like-minded users manually [34]. Systems like GroupLens [53], [79], Ringo [92] and Video Recommender [42] were the first to generate predictions using collaborative filtering algorithms. Other examples include the recommendation systems of Amazon.com, MovieCritic and Jester [35] for books, movies and jokes respectively, and the PHOAKS system which guides people to find relevant information over the web [98].

The principal difference between content-based filtering and collaborative filtering is that in the latter users' opinions are used instead of content. Thus, content-based filtering takes under consideration the descriptive features of the items while collaborative filtering utilizes users' opinions on each item. These opinions can be drawn by explicit user ratings or by implicit ratings through logging users' actions, whereas viewing/using or ignoring/skipping items could be interpreted as positive or negative ratings respectively. The opinions are used for the representation of each item. Also, text annotations attached to items could express users' opinion allowing so content-based methods to be applied as well.

We may consider collaborative filtering in recommendation process as a social activity trying to automate word-of-mouth recommendations: the items to be recommended come from other people with similar taste [92]. Therefore the principal effort of collaborative filtering concentrates on identifying users having relevant preferences and interests by calculating similarities and dissimilarities between user profiles [40]. The idea behind this is that someone who searches for information may benefit from consulting the behavior of other users that share relevant or even same interests and whose opinion can be trusted.

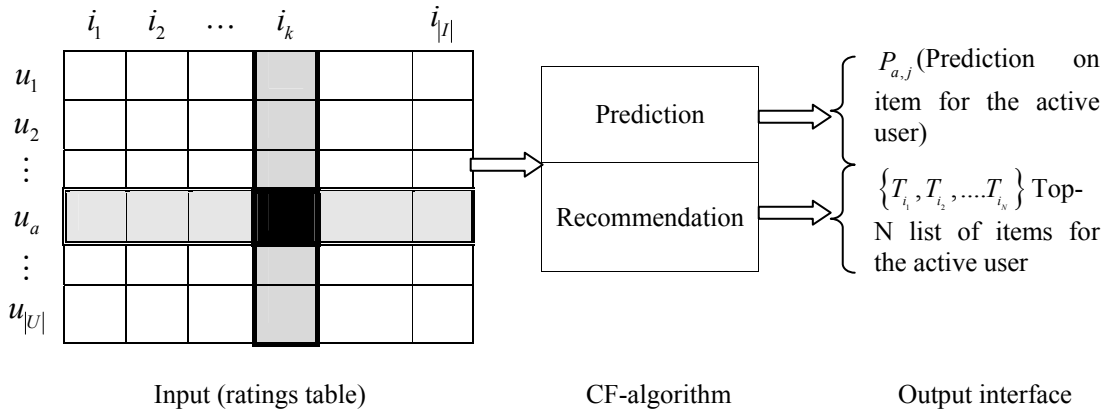


Figure 2-2: The collaborative filtering process

Figure 2-2 shows the schematic diagram of the collaborative filtering process. CF algorithms represent the entire $|U|\times|I|$ user-item data as a ratings matrix, R.Each entry $r_{i,j}$ in R represents the preference score (ratings) of the i^{th} user on the j^{th} item. Each individual rating is within a numerical scale and it can as well be 0 indicating that the user has not yet rated that item. Researchers have devised a number of collaborative filtering algorithms divided by [14] into two main categories – *Memory-based* and *Model-based* algorithms. In the following sections more detailed analysis of this categorization is provided.

2.4.2.1 Model Based Algorithms

Given any information about the user, the collaborative filtering task can be considered from a probabilistic point of view as calculating the expected value of a rating. The goal is to predict the ratings for the unobserved items. Assuming that rating values range from 0 to a highest value r the expected value of the rating the active user a will to an item j based on previously rating history:

$$E\left(r_{a,j}\right)=\sum_{i=0}^{|r|} i\cdot \Pr(r_{a,j}=i \mid r_{a,k}, k\in I_a) \tag{2.1}$$

or the probability distribution for each possible rating i:

$$p_{a,j} = \Pr(r_{a,j} = i \mid r_{a,k}, k \in I_a) \tag{2.2}$$

where I_a is the set of active user's observed ratings. There follows a brief description of the two alternative probabilistic models for collaborative, Bayesian clustering and Bayesian networks filtering.

2.4.2.1.1. Bayesian Clustering

This model makes use of a Bayesian classifier and is based on the idea that there exist certain types or groups of users appearing to have a common set of tastes and preferences. Likeminded users are clustered into classes C. Given membership to a class the preferences expressed in ratings to a variety of items are independent. The probability model used to compute the observation of an individual of a particular class given the rating profile of all users is the standard "naïve" Bayes model:

$$\Pr(C = c \mid u_1 \ldots u_n) \propto \Pr(C = c) \cdot \Pr(u_1 \mid C = c) \ldots \Pr(u_n \mid C = c) \tag{2.3}$$

The number of classes and parameters as is the probability of class membership $\Pr(C = c)$ and the conditional probability of user ratings given the class $\Pr(u_i \mid C = c)$ are learned by training the user database. Once the parameters are estimated, predictions for user ratings can be made by either computing the probability distribution of all possible ratings x of active user a to item j (Equation (2.4)) or estimating the rating using the average rating of all users in a cluster on the item (Equation (2.5)):

$$p_{a,j} = \sum^{|C|} \Pr(r_{a,j} = x \mid a \in c) \cdot \Pr(a \in c) \tag{2.4}$$

$$E(r_{a,j}) = \sum^{|C|} \Pr(a \in c)(avg(j)) \tag{2.5}$$

2.4.2.1.2. Bayesian Networks

Another probabilistic collaborative filtering approach is that of Bayesian networks. Based on this model each item in the domain is represented as a node in the network, where the states of each node refer to the possible rating values for each one item, including a state corresponding to "no vote" in order to indicate domains where data are missing.

The learning algorithm is applied to the training data which seeks after dependencies for each item over various model structures. This results in a network where each item has its best rating predictors as a set of parent items. Then the conditional probabilities for each node are encoded using decision trees as the one represented in Figure 2-3.

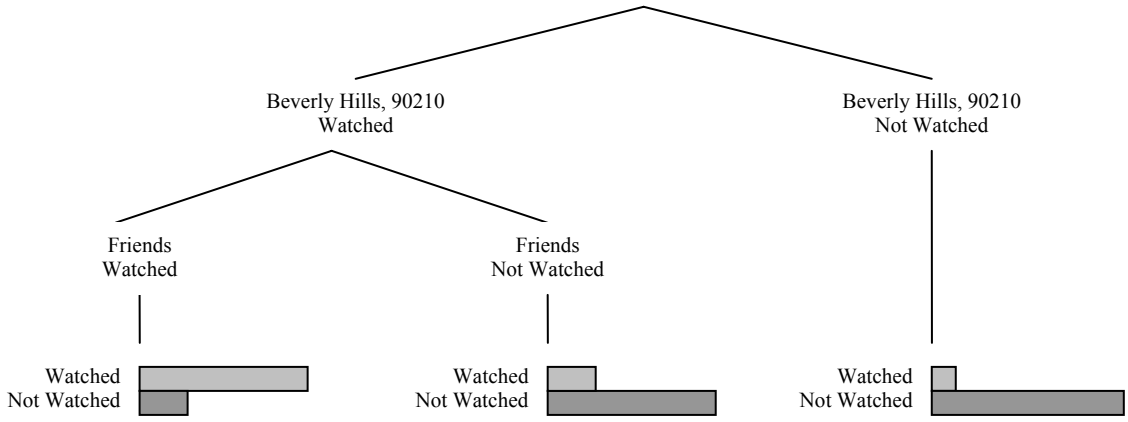


Figure 2-3: A decision tree for whether an individual watched "Melrose Place", with parents "Friends" and "Beverly Hills, 90210". The bar charts at the bottom of the tree indicate the probabilities of watched and not watched for "Melrose Place", conditioned on viewing the parent programs.

Nodes can represent any kind of variable, be it a measured parameter, a latent variable or a hypothesis. They are not restricted to representing random variables. This is what is "*Bayesian*" about a Bayesian network. If there is an arc from node A to another node B, then variable B depends directly on variable A, and A is called a parent of B. If for each variable X_i, $i = 1...N$, the set of parent variables is denoted by $parents(X_i)$ then the joint distribution of the variables is product of the local distributions:

$$\Pr(X_1,\ldots,X_n)=\prod_{i=1}^{n}\Pr\left(X_i \mid parents\left(X_i\right)\right) \tag{2.6}$$

While other recommender applications benefit from grouping users into several categories at once, this approach is limited in clustering each user into a single cluster. This results in providing recommendations only for items of the category where user belongs since items from other categories will be excluded. For example, in a book recommendation application, a user may be interested in one topic (e.g., architecture) for work purposes and a completely different topic (e.g., gardening) for leisure.

2.4.2.2 Memory Based Algorithms

Memory-based algorithms [14], [26], [63], [79], [92] utilize all information provided by users in terms of ratings to generate a prediction. These algorithms are essentially heuristics employing statistical techniques to find a set of users, known as neighbors, that have a history of agreeing with the target user (i.e., they either rate different items similarly or they tend to buy similar set of items). Once a neighborhood of users is formed, these systems use different algorithms to combine the preferences of neighbors to produce a prediction or top-N recommendations for the active user. The techniques, also known as nearest-neighbor or user-based collaborative filtering, are more popular and widely used in practice.

Thus, memory-based algorithms compute the value of the unknown rating $r_{u,i}$ for user u and item i by aggregating the ratings of other (usually the N most similar) users on the same item i:

$$r_{u,i}=\underset{u'\in\tilde{U}}{aggr}\, r_{u',i} \tag{2.7}$$

where $\tilde{U}$ is the set of the N most similar users u' to user u that have rated item i (N ranges from 1 to the number of all users). The aggregation above can be a simple averaging scheme as:

$$r_{u,i}=\frac{1}{N}\sum_{u'\in\tilde{U}} r_{u',i} \tag{2.8}$$

$$r_{u,i} = k \cdot \sum_{u' \in \bar{U}} sim(u,u') \times r_{u',i} \tag{2.9}$$

$$r_{u,i} = \overline{r_u} + k \cdot \sum_{u' \in \bar{U}} sim(u,u') \times \left(r_{u',i} - \overline{r_{u'}} \right) \tag{2.10}$$

The most widely accepted aggregation method is the calculation of the weighted sum as in Equations (2.9), (2.10). In these, $sim(u,u')$ is a heuristic-based function to calculate similarity between users u and u'. It essentially measures geometric distance of users in the space of items and is used as a weight in the prediction step. Thus, the more similar a user u is to active user u' the more emphasis will be given on his rating $r_{u,i}$ in the prediction of $r_{u',i}$. Normalizing factor k is computed as follows:

$$k = \frac{1}{\sum_{u' \in \bar{U}} sim(u,u')} \tag{2.11}$$

The problem in Equation (2.9) is that it uses absolute ratings, ignoring the fact that users rate differently. The adjusted weighted sum, as in equation, that takes under consideration each user's deviation from his average rating is a solution to this limitation. Given the set of user's u ratings $I_u = \{ i \in I \mid r_{u,i} \neq \varnothing \}$, his average rating $\overline{r_u}$ is defined as:

$$\overline{r_u} = \frac{1}{|I_u|} \sum_{i \in I_u} r_{u,i} \tag{2.12}$$

While all efforts were focused in relationships between users, in [84] the same techniques were proposed for computation of similarities but between *items* instead of obtaining their ratings from similar users. Furthermore, [84] presents empirical evidence that item-based algorithms can provide better computational performance than traditional user-based collaborative methods, while at the same time also providing better quality than the best available user-based algorithms.

Memory-based algorithms could be considered as a dual technique depending on whether predictions derive from similar users or similar items. Hereupon ensues a short presentation of the two alternative approaches.

2.4.2.2.1. User Based Collaborative Filtering

The Ringo [92] and GroupLens [53] projects have been examples of the first recommender systems that made use of user-based collaborative filtering techniques. Each user u is represented as a vector $\vec{u}$ in the $|I|$-*dimensional* space by his ratings and, then, common statistical correlation coefficients (Pearson correlation [79]) or the cosine similarity measure (from information retrieval [4]) can be employed to compute similarities $sim(u,u')$ between all pairs of users $(u,u') \in U \times U$. Notice that all user-user correlations $sim(u,u')$ are computed only over the subset of items that both users have rated, i.e $I_{u,u'} = \{i \in I \mid r_{u,i} \neq \perp \cap r_{u',i} \neq \perp\}$.

As already expected by its name, the cosine similarity measure aims at quantifying the similarity between user-vectors $\vec{u}, \vec{u'}$ by computing the cosine of their angles:

$$sim(u,u') = \cos\left(\vec{u},\vec{u'}\right) = \frac{\vec{u} \cdot \vec{u'}}{\left\|\vec{u}\right\|_2 \cdot \left\|\vec{u'}\right\|_2} = \frac{\sum_{i=0}^{|I_{u,u'}|} r_{u,i} \cdot r_{u',i}}{\left(\sum_{i=0}^{|I_{u,u'}|} r_{u,i}^2 \cdot \sum_{i=0}^{|I_{u,u'}|} r_{u',i}^2\right)^{\frac{1}{2}}} \tag{2.13}$$

Pearson correlation comes from a linear regression model [38] and is similar to cosine similarity but takes account of users' deviation from their mean rating. Given the averages $\bar{u}, \bar{u'}$ the similarity of users u, u' is calculated as:

$$sim(u,u') = \frac{\sum_{i=0}^{|I_{u,u'}|} \left(r_{u,i} - \bar{r_u}\right) \cdot \left(r_{u',i} - \bar{r_{u'}}\right)}{\left(\sum_{i=0}^{|I_{u,u'}|} \left(r_{u,i} - \bar{r_u}\right)^2 \cdot \sum_{i=0}^{|I_{u,u'}|} \left(r_{u',i} - \bar{r_{u'}}\right)^2\right)^{\frac{1}{2}}} \tag{2.14}$$

Since similarities $sim(u,u')$ between all pairs $(u,u') \in U \times U$ have been computed using either the cosine similarity measure or Pearson correlation coefficient, neighborhoods $prox(u)$ of the *top-N* most similar neighbors are being formed for every user $u \in U$. The next step is to predict the rating for all items p_i rated by u's neighbors but yet unknown to u, i.e, more formally, predictions $w_i(p_i)$ for $p_i \in \{i \in I \mid \exists u' \in prox(u) : r_{u',i} \neq \varnothing \cap r_{u,i} = \varnothing\}$:

$$w_i(p_i) = \overline{r_u} + \frac{\sum_{u' \in prox(u)} \left(r_{u',i} - \overline{r_{u'}} \right) \cdot sim(u,u')}{\sum_{u' \in prox(u)} sim(u,u')} \tag{2.15}$$

As discussed before, predictions are based upon adjusted weighted sum of u's neighbors' ratings. Finally, a list of *top-N* recommendations $P_{w_i} : \{1,2,\ldots,N\} \to I$ is computed in a descending order, based upon predictions w_i and giving priority first to highest predictions.

Performance Tuning

In order to tune the performance of collaborative filtering several methods have been proposed:

- **Inverse user frequency:** In information retrieval, [4] used the "*inverse document frequency*" to modify word frequencies in applications based on vector similarity. The key idea was to reduce weights for frequently occurring words when computing similarities between documents, considering that less common words are more significant in search of a topic. An analogous approach, termed "*inverse user frequency*", was adopted by [14] in the notion that universally liked items may not be as indicative of similarity as less favorited items. Hence, co-votes on less popular items are much more rewarded than co-votes on very popular items.

- **Significance weighting:** As earlier described, similarities between users in collaborative filtering are computed over the set of their co-rated items. Suppose that two users have only one item in common. In this case, their correlation gets its maximum value if they give same ratings. It's obvious that collaborators derived from cases similar to the one above are not trustworthy. In [38] a threshold for the number of co-rated items has been introduced. By applying a weight of c/t, where c is the number of common ratings and t the threshold ($t = 50$ in [38]), user correlations based upon co-rated items fewer than the threshold are being punished. [14]

proposed "*default voting*" as another approach to alleviate the same problem to filling the "*missing votes*".

- **Case amplification.** While both previous modifications refer to the process of similarity computation, "*case amplification*" [14] handles the rating prediction step by giving emphasis on high correlation weights $sim(u,u')$ and penalizing low correlation weights, as in Equation (2.16):

$$sim'(u,u') = \begin{cases} sim(u,u')^{\rho} & \text{if } sim(u,u') \geq 0 \\ -(-sim(u,u'))^{\rho} & \text{else} \end{cases} \quad (2.16)$$

Thus, users with high degree of similarity will have much more impact in rating prediction than before.

Information Filtering Agents

Researchers, as in [36], [86] injected the concept of *filterbots* (term derived from "*filtering robots*") in the sphere of user-based collaborative filtering. Filterbots are automated rating robots which participate as members of a CF system. They help users who agree with them by providing more ratings upon which recommendations could be made.

For example, in GroupLens Usenet news recommender system [53], filterbots were crewed to rate Usenet articles focused on spelling errors, text length, and so on. It was shown in [86] that the use of filterbots could lead to improved recommendation accuracy when operating in sparse data enviroments.

2.4.2.2.2. Item Based Collaborative Filtering

Item-based CF [27], [50], [84] has been gaining momentum over the last few years because it decouples the process of model computation from the entire prediction procedure allowing so to scale on large data sets. Specifically, in cases where $|U| \gg |I|$, item-based CF has shown to perform better than traditional user-based CF

[69], [70], [84]. Many commercial recommender systems, including Amazon.com [56] and TiVO [2], adopted this approach to benefit from its virtue.

Alike user-based CF, recommendations are based upon ratings $r_{u,i}$ that users $u \in U$ provide for items $i \in I$. However, in contrast with user-based CF, similarity computations are carried out over items instead of users, hence $sim : I \times I \to [-1,+1]$. Roughly speaking, two items i, i' tend to be similar, i.e. have large $sim(i,i')$, if users who rated one of them tend to rate the other and the ratings they assign tend to be similar or identical. Effectively, item-based similarity computation equates to the user-based case when rotating the user-item matrix by 90°. Next, for each item i neighbourhoods $prox(i) \subseteq I$ are formed including the top-N most similar items i'. To compute the predictions $w_i(p_i)$ for all the items $p_i \in \{i' \in I \mid \exists i' \in prox(i) : r_{u,i'} \neq \varnothing \cap r_{u,i} = \varnothing\}$ the following equation is used:

$$w_i(p_i) = \frac{\sum_{i' \in prox(i)} r_{u,p_i} \cdot sim(i,i')}{\sum_{i' \in prox(i)} sim(i,i')} \tag{2.17}$$

Intuitively, item-based approach can be stated as follows: given a user u and his known ratings, try to predict the value of an unknown item i_k by comparing the latter to known and similar items, considering the degree user u likes them.

Next, as with user-based CF, a list of top-N recommendations $P_{w_i} : \{1,2,\ldots,N\} \to I$ is computed in a descending order, based upon predictions w_i and giving priority first to highest predictions.

2.4.3 Demographic-based Techniques

In demographic recommender systems users are classified by their personal attributes and recommendations rely on demographic classes. *Grundy* [81] is an example of this type of recommender. This system uses an interactive dialogue to gather personal data and, based upon this information, provided recommendations on books. User-stereotypes were manually assembled into a library where responses collected by real time users were matched against them. Some more recent

recommender systems also adopted this approach. For instance, [54], conducts marketing research on demographic groups in order to suggest a range of products and services. User data for categorizing are collected by a short survey while other systems, such as [71], use a learning machine to classify over the demographic data. Various ways can be used to represent demographic information in a user model. Rich's system uses hand-crafted attributes taking numeric confidence values. Pazzani's model extracts features from users' home pages that are indicative of liking certain restaurants. Demographic techniques are similar to collaborative filtering in the way they form "people-to-people" correlations. The only difference is in the nature of data they use. The main benefit of the demographic approach is the ability to provide recommendations without a history of user ratings of the type required by collaborative and content-based techniques.

2.4.4 Utility-based Techniques

Utility-based recommenders attempt to match user's need against a set of available options rather instead of building long-term generalizations about users. Therefore these systems base their advice on the utility computation of each object to the user. As expected, the main problem here lies in the creation of a utility function for each user. *Tête-à-Tête* and the e-commerce site *PersonaLogic* employ different techniques, each, to arrive at a user-specific utility function that will later use all over the objects [37]. Therefore, user's profile is what systems derives and operates as a utility function along with constraint satisfaction techniques to find the best match. The benefit of utility-based recommendation is that it can enbody non-product attributes related to the value of a product into the utility computation, e.g. product availability and vendor reliability, allowing so a trade off between product price and delivery schedule to a user who is, for example, in urgency.

2.4.5 Knowledge-Based Filtering

Knowledge-based recommender systems propose objects relying on inferences about a user's preferences end needs. This is something common in all recommendation systems but the only difference with knowledge-based approaches is

the use of functional knowledge regarding user's preferences and needs. Therefore, these systems intend to reason the relationship between a need and a potential recommendation, since "knowing" how a particular product meets a user's needs. The knowledge to build user profile can use any structure supporting this inference.

The structure to be used can be of any form. The simplest form can be a query a user just formulated, like in Google. Other cases may include more detailed user representation [99]. Case-based techniques for knowledge-based recommendation have been employed by the *Entrée* system [17], [18] and many other recent systems, such as in [88], [90] consider the "*Editor's choice*" method as a knowledge-based technique.

Furthermore, the content of knowledge used in recommendation systems of this category can take a variety of forms. Google, for instance, utilizes information about the links between web pages to examine whether they are popular, authoritative [15]. Knowledge regarding cuisines is used by Entrée to discover similarities among restaurants. However, existing systems require users to map their needs against product features by answering to detailed questionnaires, as in *PersonaLogic*, or using preference functions for each feature, as in *Tête-à-Tête*.

2.4.6 Economic Filtering

Economic filtering takes the notion of suggesting products somewhat further by introducing cost into the recommendation procedure. The main idea now is not to filter users or products, as in previous recommendation methods, but to encourage sending recommendations. In this concept, users could be awarded for sending recommendation or charged for receiving recommendations. This could be achieved by using some kind of reward or other mean. In the simplest case, charging could be analogous to the length of the recommendation.

This kind of filtering is not a very popular method in recommender systems. A system that makes use of virtual money as reward and payment for sending and giving recommendations respectively is *Knowledge Pump* in [32].

2.4.7 Summary

As discussed earlier in this section, recommendation systems can be categorized depending on the following criteria: the nature of information (as a background and an input) they utilize and the methods (as a process) they adopt to this end.

Technique	**Background**	**Input**	**Process**
Collaborative	Ratings from U of items in I.	Ratings from u of items in I.	Identify users in U similar to u and extrapolate from their ratings of i.
Content-based	Features of items in I.	u's ratings of items in I.	Generate a classifier that fits u's rating behavior and use it on i.
Demographic	Demographic information about U and their ratings of items in I.	Demographic information about u.	Identify users demographically similar to u and extrapolate from their ratings of i.
Utility-based	Features of items in I.	A utility function over items in I describing u's preferences.	Apply function to items and determine i's rank.
Knowledge-based	Features of items in I. Knowledge of how these items meet a user's needs.	A description of u's needs or interests.	Infer a match between i and u's needs.
Economic	Any information provided by users of U for items in I.	Recommendation from u for items in I.	Users u are rewarded for sending recommendations and charged for receiving recommendations.

Table 2-3: Summary of recommendation techniques.

Let U, I denote the sets of users and items, and u, i indicate an individual user item. Table 2-3 summarizes all the recommendation techniques described in this section.

2.5 Limitations in Recommender Systems

Recommendation techniques have been successfully adopted by research and industrial applications in order to filter out and personalize information according to user interests. However, these methods encompass certain shortcoming that may have a negative impact on the effectiveness of the recommendation algorithms. In this

section we describe these shortcomings and discuss recent research that has been conducted to get over them.

2.5.1 Limited Context Analysis

Content-based techniques are limited by the features explicitly associated with the objects that these systems recommend. Therefore, in order to have a sufficient set of features, the content must either be in a form that can be parsed automatically by a computer (e.g., text), or the features should be assigned to items manually. While information retrieval techniques work well in extracting features from text documents, some other domains have an inherent problem with automatic feature extraction. For example, automatic feature extraction methods are much harder to apply to the multimedia data, e.g., graphical images, audio and video streams. Moreover, it is often not practical to assign attributes by hand due to limitations of resources [92].

Another problem with limited content analysis is that if two different items are represented by the same set of features, they are indistinguishable. Therefore, since text-based documents are usually represented by their most important keywords, content-based systems cannot distinguish between a well-written article and a badly written one, if they happen to use the same terms [92].

2.5.2 Over-specialization

In the content-based approaches, when the system can only recommend items that score highly against a user's profile, the user is limited to being recommended items similar to those already rated. For example, a person with no experience with Greek cuisine would never receive a recommendation for even the greatest Greek restaurant in town. This problem, which has also been studied in other domains, is often addressed by introducing some randomness. For example, the use of genetic algorithms has been proposed as a possible solution in the context of information filtering [93]. In addition, the problem with over-specialization is not only that the content-based systems cannot recommend items that are different from anything the user has seen before. In certain cases, items should not be recommended if they are

too similar to something the user has already seen, such as different news article describing the same event.

Therefore, some content-based recommendation systems, such as DailyLearner [12], filter out items not only if they are too different from user's preferences, but also if they are too similar to something the user has seen before.

2.5.3 Scalability

Collaborative Filtering seems to be efficient in filtering in items that are interesting to users. However, it requires computations that are very expensive and grow nonlinearly with the number of users and items in a database. Therefore, in order to bring recommendation algorithms successfully on the web, and succeed in providing recommendations with acceptable delay, sophisticated data structures and advanced, scalable architectures are required. In [24], authors describe an open framework for practical testing of recommendation systems in an attempt to provide a standard, public test bed to evaluate recommendation algorithms in real-world conditions.

The collaborative method generates recommendations based on a subset of users that are most similar to the active user. The formulation of a single recommendation is a two-step computation. First, the algorithm needs to compute the similarity between the active user and all other users, based on their co-rated items, so as to pick the ones with similar behavior. Subsequently, the algorithm recommends to the active user items that are highly rated by his or her most similar users. In order to compute the similarities between users, a variety of similarity measures have been proposed, such as Pearson correlation, cosine vector similarity, Spearman correlation, entropy-based uncertainty measure and mean-square difference.

Collaborative filtering fails to scale up its computation with the growth of both the number of users and items in the database. To deal with the scalability problem [14] and [102] utilize Bayesian network and clustering approaches, while [84] apply folding in *Singular Value Decomposition* (SVD) to reduce the dimensionality of the user-item matrix. It is also possible to address these scaling issues by data reduction or data focusing techniques. [108] and [109] adopt instance selection for removing the irrelevant and redundant instances. Moreover, content-boosted Collaborative Filtering approaches reduce the number of items examined, by partitioning the item space

according to item category or subject classification. Similar approaches compress user data by clustering techniques and then apply CF either over the clusters generated [78] or over users located in the same cluster [107]. Such techniques that require a preprocessing step are discussed in detail in next section (Metal-level hybrids). Finally, more greedy approaches concentrate on randomly sampling users, discarding users with few ratings or discarding very popular or unpopular items.

Unfortunately, even when these methods achieve improved performance, they also reduce recommendation quality in several ways. Bayesian networks may prove practical for environments in which user preferences change slowly with respect to the time needed to build the model, but are not suitable for environments in which user preference models must be updated frequently. Clustering-based methods suffer from poor accuracy. It is possible to improve their quality by using numerous fine grained segments [49], but then online user segment classification becomes almost as expensive as finding similar users using the classic Collaborative Filtering. SVD-based work focuses mainly on accuracy rather than efficiency. Data focusing and reduction approaches, such as instance selection or item-space partitioning, experience reduced accuracy due to loss of information. If an algorithm discards the most popular or unpopular items, there may be items that will never be recommended to some users. Obviously, to gain in computation one needs to lose in recommendation quality and vice versa. Appropriate trade-offs must be considered.

Finally, in [70] it has been suggested that similarities of all possible pairs of users should be placed in a memory cache while effort focused on incremental update of the stored information. However, an essential point as memory complexity is not addressed. The memory complexity induced by the caching of data in this approach seems quite problematic especially for large scale problems because the recommender system needs to process a huge amount of data.

2.5.4 Sparsity

Several principled methods have been proposed to address the data sparsity problem which refers to statistic-based methods, i.e. collaborative filtering. Most popular approaches use item-based similarity instead of user-based similarity [84],

dimensionality reduction of the user-item matrix, application of associative retrieval techniques, and content-boosted collaborative filtering.

Dimensionality reduction techniques address the sparsity problem by removing non representative or insignificant users or items in order to condense the user-item matrix. More advanced techniques for dimensionality reduction have been proposed in as well. Examples include statistical techniques such as *Principle Component Analysis* (PCA) [35] and information retrieval techniques such as *Latent Semantic Indexing* (LSI) [25], [43], [11]. [109] proposed to compute the users' similarity by a matrix conversion method for similarity measure. So, instead of computing user similarities over the traditional user-item matrix, a naïve Bayesian classifier was applied to the creation of a user-class matrix. However, as shown in [45], [107] potentially useful information might be lost during the reduction process described above.

Content-boosted CF [7], [21], [59] approaches require additional information regarding items as well as metrics to compute meaningful similarities among them. In [75] a unified probabilistic model for integrating content information was proposed to cope with sparse data environments. However, such information about items may be difficult or expensive to acquire in practice. More hybridization strategies that combine content-based and CF techniques are further described in next sections.

Another recent direction in collaborative filtering research is by combining memory-based and model-based approaches [73], [107]. For example, [107] formed clusters of users and after applying cluster-based smoothing they used them as basic units in making recommendations. As pointed out in [78], such methods could hurt the coverage of the recommender for two main reasons: first, users from the picked cluster may have rated a small fraction of the items available than people in other clusters and, second, an important fraction of closest neighbors may reside in other than selected cluster.

Recently, [45] have investigated the impact of transitive associations between users and items. This approach however failed to express the subjective notion of associations. To solve this problem [33], [68], [74] proposed a computational trust model based on trust inferences. In a context of social network, such methods maybe ineffective due to increased computational cost in applying transitive properties between users.

[104] has extended the probabilistic relevance model used in text retrieval ([41]) to the problem of collaborative filtering by adopting a linear interpolation smoothing. These approaches were limited to binary rating data so, later on, [105] reformulated collaborative filtering into a generative probabilistic framework treating user-item ratings individually as predictors for missing ratings.

2.5.5 Cold Start

The term *cold-start* is used to emphasize the importance of the sparsity problem observed in the learning-based techniques, such as content-based, demographic-based and collaborative filtering. The cold-start problem [89] describes the situation in which an item cannot be recommended due to deficiency in raters. This problem refers specifically to new and obscure items and is particularly detrimental to users with eclectic taste. Likewise, a new user has to provide a sufficient number of ratings so that the recommendation algorithm is able to make reliable and accurate recommendations.

2.5.5.1 New Item Problem

New items are added frequently to recommendation systems. Collaborative systems are based solely on users' preferences in order to provide recommendations. Therefore, by the time that the new item has a substantial number of ratings, the recommendation system would not be able to recommend it. The most common way to address this problem is by using hybrid recommendation approaches.

2.5.5.2 New User Problem

A system must first learn about user's preferences from his rating activity in order to make accurate recommendations. To cope with this problem several techniques have been proposed. Most of them are based on hybridization techniques which combine content-based and collaborative techniques. In [77] an alternative approach explores various techniques to determine the best items for a new user to rate, i.e. the most informative items to a recommendation system. These techniques make use of

strategies based on item popularity, user personalization, item entropy, and also combinations of the above [77].

2.5.6 Learning Inertia

Utility-based and knowledge-based recommenders do not have cold-start problems since they do not rely on accumulated statistical evidence to provide recommendations. However, since they are not based on statistics, which are provided by other users' transactions in the system, such methods have static learning ability. For that, utility-based techniques require the existence of a complete utility function across all features of the items the system possesses. A utility-based framework thereby lets the user express all of the considerations that need to go into a recommendation.

Moreover, utility-based systems fail to some degree in flexibility. The user must construct a complete preference function, and must therefore weigh the significance of each possible feature. Often this introduces a significant burden of interaction. This could make a utility-based system inappropriate for a casual browser.

2.5.7 Summary

In this section some of the major problems that recommender techniques suffer from have been discussed along with some proposals to their solution. For each recommender technique its strengths and its weaknesses are summarized in Table 2-4.

Technique	Pluses	Minuses
Collaborative	**A.** Ability to identify cross-genre niches. **B.** Domain knowledge not required. **C.** Adaptability: quality improves over time. **D.** Sufficiency of implicit feedback.	**J**. New user problem **K**. New item problem **L**. "Gray sheep" problem (no matching with users' cliques) **M**. Quality dependency on historical data set.
Content-based	**B, C, D**	**J**, **M**
Demographic	**A, B, C**	**J**, **L**, **M**, **N**.Must collect demographic information.
Utility-based	**E.** No cold-start **F.** Sensitivity to changes of preference **G.** Can include non-product features	**N**. Utility function input required **O**. Static ability in suggesting (system does not learn)
Knowledge-based	**E, F, G,** **H.** Ability to map user needs across products.	**O,** **P**. Knowledge engineering is required to identify associations ahead of time.
Economic	**I.** No computational cost required.	**Q.** Attractive means to make users provide recommendations

Table 2-4: Summary of trade off in recommendation techniques.

2.6 Hybrid Methods

Hybrid recommender systems are combinations of two or more recommendation techniques aiming at better performance with fewer drawbacks than any individual one. Most commonly, collaborative filtering is combined with some other technique to avoid the cold start problem.

2.6.1 Weighted

In a weighted hybrid recommender the final score of a specific item is calculated combining the results derived from all recommendation techniques available in the system. In the simplest case, such a hybrid could be a linear combination of the scores which all "components" of the recommendation system produce. In [21] the P-Tango system is an example of this case. First it gives both collaborative and content-based recommenders same weight, but gradually and as predictions on user's ratings are confirmed or disconfirmed the weighting is adjusted properly. Pazzani's system [71] is another example of combination hybrid which uses a consensus scheme to combine

the results of each recommender (content-based, collaborative and demographic) treating them rather as a set of votes instead of numeric scores. The main benefit derived from the hybrids of this category is that all of the system's constituents operate independently and stand in a straightforward way over the recommendation process. So, it is easy to adjust the hybrid performing post-hoc credit assignment. However, in this technique it is implicitly assumed that the different techniques have more or less the same utility across the space of possible items. From what we discussed in previous sections, it is obvious that this is not always so: a collaborative recommender is weak when an item has only few raters but gets stronger as the number of ratings grows.

2.6.2 Switching

As expected by the name, hybrids of this category use some kind of criterion to switch from one recommender technique to another. An example of this hybridization technique is the *DailyLearner* system [12] where content and collaborative methods are employed together keeping the content-based recommendation method as a leader. When there is not enough confidence in the result of the content-based system, recommendations are based upon collaborative filtering.

Actually Billsus' system has one short-term and one long-term content-based recommendation algorithm and follows the fallback strategy "short-term – collaborative – long-term". Thus, the short-term method is used first and when the technique fails then the next one in row comes to continue. [101] proposed a more straightforward switching hybrid. In their system, the technique to be selected for the next recommendation depends on the agreement of user's past ratings with the recommendations provided by each component.

Since both content-based and collaborative-based suffer from the "new user" problem, the same will hold for their hybridization. However, *DailyLearner*'s content-based technique uses nearest-neighbor heuristics where large number of examples is not required for accurate classification.

Thanks to collaborative filtering a switching hybrid is able to cross genres so that recommendations provided are not "*semantically*" close to the items previously rated high but still relevant. The use of switching hybrids leads to additional burden

because of the computational complexity needed to estimate switching criteria into the recommendation process, and this in turn to another level of parameterization. However, the benefit is that the system can be sensitive to the strengths and weaknesses of its constituent recommenders.

2.6.3 Mixed

In the mixed hybrids recommendations from all available techniques are presented simultaneously. An instance of this hybridization technique is the *PTV* system [94] which aims to create a recommended television program using content-based techniques based upon textual descriptions of TV shows and collaborative filtering over users' preferences. The final recommended program consists of a combination of the suggestions provided by the two techniques.

The mixed hybrid overcomes the "new item" start-up problem as follows: when recommending new shows, even if they have not been rated by anyone, the content-based component relies on the content description. This hybrid does not cope with the "new user" start-up problem since both content and collaborative methods need some data to get off the ground concerning user's preferences. But regarding TV program recommendations, if such a system is embedded into a digital television, it can keep a log of the shows and the time they were watched to build profiles accordingly. So, exploiting the "*niche-finding*" property of this hybrid, as in the case of switching hybrid, this technique can recommend new items that would be eliminated if focusing strictly on content.

The *PTV* system is somewhat unusual because recommendations are made to construct a composite entity, the program schedule to be viewed. Such a schedule needs loads of recommendations to be filled, so suggestions from as many as possible sources can be useful to this end. Whenever conflicts occur, a sort of arbitration between methods should be arranged – in case of *PTV*, content-based recommendations gain precedence over collaborative filtering responses. Other applications of the mixed hybrid are *ProfBuilder* [106] and *PickAFlick* [17], [19], presenting side-by-side multiple recommender sources.

2.6.4 Feature Combination

Another way to merge content and collaborative techniques is to simply associate collaborative information as additional feature with each example data and apply content-based techniques over the augmented data set. For example, Basu et all. [8] reported significant improvements in precision, compared to a purely collaborative approach, by applying the inductive rule learner *Ripper* to the task of recommending movies over both user ratings and content features. However, this benefit could only be achieved by hand-filtering contents features. The authors found that employing all of the available content features improved recall but not precision.

The fact that feature combination hybrid techniques utilize collaborative data without depending on it exclusively reduces the sensitivity of the system to the number of raters. Conversely, it allows the system to maintain and handle information about the inherent similarity of items that would be otherwise opaque to a collaborative filtering system.

2.6.5 Cascade

The cascade hybrid differs from previous hybridization methods in that a staged process is involved. So, this technique suggests the application of one recommendation technique in a first level in order to produce a coarse ranking of candidates and, afterwards, the use of a second technique to refine the recommendation coming from the candidate set. The restaurant recommender system *EntreeC* [18] is a cascaded hybrid from knowledge-based and collaborative recommenders. The knowledge about restaurants used for making recommendations is based upon user's stated interests. Recommendations derived from knowledge-based technique are placed into buckets of equal preference where the collaborative filtering technique is employed to further rank the suggestions in each bucket.

In the second step the candidate items are already well-differentiated by the first stage excluding those that will never be recommended anyhow. Thus, the system is able to apply a second, low-priority, technique onto a smaller set of items that need additional discrimination. This is more efficient than applying all component-techniques to all items available as in other hybrids - for instance the weighted hybrids. Moreover, since ratings derived by the high-priority technique cannot be

misquoted, but only refined, the cascade is proved by its nature to be tolerant of noise when operating the low-priority technique.

2.6.6 Feature Augmentation

In feature augmentation techniques one technique is employed first to produce a rating or a classification of an item and the result is then used by the next recommendation technique. For instance, the *Libra* system [62] makes books recommendations using a content-based technique – specifically, a naïve Bayes text classifier was used – on data found in Amazon.com. The system uses text data that contains information about "*related authors*" and "*related titles*" generated by collaborative systems from Amazon. It is found that these features improve the quality of recommendations.

Feature augmentation techniques were also employed by the *GroupLens* research team in *Usenet* news filtering [86]. In this instance, to improve recommendation quality they made use of *filterbots* – a fair description of filterbots lies in previous section.

Feature augmentation offers the capability of improving a core system's performance without having to modify it. Examples of such systems are the *NetPerceptions*' *GroupLens Recommendation Engine* or even a naïve Bayes' text classifier. Additional functionality is added by intermediaries who can use other techniques to augment the data itself. Augmentation differs from feature combination in that raw data coming from different sources is to be combined.

Although both cascade and augmentation techniques use two recommenders in sequence, with the first recommender having an influence over the second, they are fundamentally different. In an augmentation hybrid, the output of the first recommender is used as an input itself for the second recommender, as in the case of ratings provided by *GroupLens' filterbots*. In a cascaded hybrid, there is no output/input but the results of the two recommenders, given priority to one of them, are somewhat combined.

2.6.7 Meta-level

Another way to combine two recommendation techniques is to use the model generated by one as input for the other. In the meta-level hybrid the entire model learnt by the first algorithm is used as input to a second algorithm while in the augmentation hybrid only features generated by the learnt model are used to this end. This is the primary difference between meta-level hybrids and the feature augmentation.

The first meta-level hybrid was the *Fab* system [5], [6], [7] where user-specific selection agents performed content-based web filtering using Rocchio's method [82] to describe user's area of interest. These agents collected documents over the web on a common basis of interest to the overall community and then distributed them to particular users. *Fab* also performed a cascade of collaborative filtering and content-based collection because in the collaborative step the rating information was used only to create a coarse pool of documents that was further processed into a content-based selection component.

Another meta-level hybrid, known as "*collaboration via content*" [71], made use of Winnow [57] to build content-based model for each user describing the features of restaurants that user potentially likes. More recently, [22] have applied the following two-stage Bayesian scheme: a content-based naive Bayesian classifier was used for each user to estimate the parameters that were further linked regressively across different users. *LaboUr* [91] used instance-based learning technique to create user profiles which were in turn compared by collaborative filtering.

The meta-level method, especially for the case of content/collaborative hybrid, has the benefit that the model learnt by the content-based mechanism constitutes a compressed representation of user interests, and so the collaborative mechanism that follows can operate on this dense representation of information more easily than on raw user data.

2.6.8 Summary

In this section we present all well-known hybridizations that were employed in order to deal with common limitations associated with any pure application of the recommendation techniques. Generally, most of the efforts focused on utilizing both

collaborative- and content-based heuristics. In Table 2-5 there is a summary of the combination methods.

	CF/CN	CF/DM	CF/KB	CN/CF	CN/DM	KB/CF
Weighted	P-Tango	[Pazzani 1999]	[Towle & Quinn 2000]		[Pazzani 1999]	
Switching	PTV, ProfBuilder					
Mixed	DailyLearner		[Tran & Cohen, 2000]			
Feature Combination	[Basu, Hirsh & Cohen 1998]				[Condliff, et al. 1999]	
Cascade	Fab					EntreeC
Feature Augmentation	Libra					GroupLens [1999]
Meta-level				Fab, [Condliff, et al. 1999], LaboUr		

CF=Collaborative Filtering, CN=Content-based,
DM=Demographic-based, KB=Knowlegde/Utility-based

Table 2-5: Summary of possible hybridization techniques.

As Table 2-5 shows, a lot of space remains to be fully explored, research has provided some insight into the question of which hybrid to employ in particular situations. The hybridization strategy must be a function of the characteristics of the recommenders being combined. With demographic, content and collaborative recommenders, this is largely a function of the quality and quantity of data available for learning. With knowledge-based recommenders, it is a function of the available knowledge base. We can distinguish two cases: the uniform case, in which one recommender has better accuracy than another over the whole space of recommendation, and the non-uniform case, in which the two recommenders have different strengths in different parts of the space. If the recommenders are uniformly unequal, it may make sense to employ a hybrid in which the inaccuracies of the weaker recommender can be contained: for example, a cascade scheme with the stronger recommender given higher priority, an augmentation hybrid in which the

weaker recommender acts as a "bot" contributing a small amount of information, or a meta-level combination in which the stronger technique produces a dense representation that strengthens the performance of the weaker one. In the non-uniform case, the system will need to be able to employ both recommenders at different times. A switching hybrid is a natural choice here, but it requires that the system be able to detect when one recommender should be preferred. Feature combination and mixed hybrids can be used to allow output from both recommenders without having to implement a switching criterion. More research is needed to establish the tradeoffs between these hybridization options.

Chapter 3

Combining Prediction Sources

3.1 Introduction

Nowadays most popular commercial systems use collaborative filtering (CF) techniques to efficiently provide recommendations to users based on opinions of other users about items. To effectively formulate recommendations, these systems rely either upon statistics (ratings from others users) or upon contextual information about items. Many hybrids of collaborative and content-based methods have been developed to eliminate sparsity and cold-start problems but none of them provides a unified framework to solve these problems.

The first challenge is the sparsity problem. Statistics have the benefit that they can learn from information provided by user and other users as well. Nevertheless, until they have gathered information they cannot provide accurate results and this refers to the sparsity problem. Despite the methods developed to cope with this problem it seems that both user-based and item-based methods need a start-up time so that they operate on a more dense ratings matrix.

The second challenge is that CF depends merely on either users or items to find associations. The quality of the computed associations and final predictions depends on how the information is distributed in the ratings matrix. More information may be on the user level or on the item level. For instance, in Table 3-1 user 1 have only one movie in common with all the other users, while movie 3 has more common ratings with movie 1 which is also rated by user 1. Thus, an item-based prediction for this item is expected to be more accurate than a user-based approach since the latter is based only on a few common items. On the other hand, in case of user-based we have more information about ratings of the item in question by other users. The same

stands and vice versa. Sarwar [84] reports that item-based CF is ideal for sparse environments while user-based approach seems to gain ground in dense matrices. In any case, these two approaches need to be combined to improve the accuracy in the recommendation process.

User / Movie	1	2	3	4	5
1	5	4	3	3	4
2	4	4	∅	3	∅
3	∅	5	2	1	4
4	∅	5	2	1	4
5	∅	4	5	4	4
6	∅	4	3	3	4
7	∅	3	2	3	2

Table 3-1: Rating information.

The third challenge is the user bias from rating history (statistics). For instance, any statistical-based approach cannot distinguish a pair of items that have the same history of ratings but different content features. For example, in Table 3-2 both user-based and item-based would recommend movies 3 and 4 to user 1. However movie 3 belongs to horror films and should be given the priority instead of movie 4, because user 1 prefers horror films according to his history of ratings. Besides, statistics work most of the time, not all the time: That's the beauty and the curse of the statistics.

Genre / Movie	Horror	Action
1	1	0
2	1	0
3	1	0
4	0	1
5	1	0

User / Movie	1	2	3
1	5	4	3
2	4	4	3
3	∅	4	3
4	∅	4	3
5	∅	∅	∅

Table 3-2: Movie-category bitmap and rating information

Moreover, other phenomena such as the "gray sheep", where a user cannot match with anyone of other users' cliques, and the cold-start problem, where a new item cannot be recommended due to the lack of any information, reduce the strength of statistic-based methods. On the other hand, methods relying exclusively on the content tend to recommend to user items similar to those already rated by a user.

All these observations advocate the any information available to the formulation of recommendations. To exploit information from different sources we use knowledge from decision theory and theory of probabilities. This framework is purely probabilistic introducing the concept of uncertainty w.r.t. the accurate knowledge of the model.

So far, any work done in this area (described in Section 2.6.1), the key idea was the linear combination of different recommendation sources [21][71][104][105], i.e. naïve combination of numerical scores derived from each source. These methods generally are not capable of providing explanations of predictions or further insights into the data. Our approach differs from all others in that now combination has a pure and meaningful probabilistic interpretation which may be leveraged to explain, justify and augment the results.

Specifically, two basic pure–probabilistic combination schemes are under examination: the sum–rule and the product–rule. The information sources used in our tests include: user ratings, item ratings and the categories they belong. Results show that pure probabilistic schemes perform better than any numerical combination of the individual methods.

3.2 Methodology

Many approaches for CF have been previously proposed. Most of them make use of heuristics as *k-nearest neighbor* (KNN), which requires the existence of common ratings between users in order to calculate similarity measures. Thus, users with no common items will be excluded from the prediction procedure. This might have a serious impact upon the coverage of the recommendation, i.e., the number of items for which the system is able to generate personalized recommendations might get lower.

Pennock [73] proposed a hybrid method combining the strengths of both model-based and memory-based approaches where it was also demonstrated that this

hybridization leaded to better recommendation results than pure memory-based and model-based approaches. This approach, named *Personality Diagnosis* (PD), is based on a simple and reasonable probabilistic model of how people rate titles. Like other model-based approaches, its assumptions are explicit, and, as told above, its results have a meaningful probabilistic interpretation. Like any other memory-based approaches it is fairly straightforward, operates over all data while no compilation step is required for new data. According to Pennock, PD makes better predictions than other well known algorithms, pure memory-based like Pearson correlation coefficient (PCC) and vector similarity (VS) as well as pure model-based like Bayesian clustering (BC) and Bayesian networks (BN).

In the following sections we describe the PD algorithm, moreover extend this model to an item-based and a content-based level while, later on, we present the two pure probabilistic combination schemes.

3.2.1 Personality Diagnosis

According to Pennock [73], PD posits that each user u_i, where $i = 1, 2 \ldots m$, given any rating information over the objects available, has a *personality type* which can be described as:

$$R_{u_i}^{true} = \left\{ r_{i,1}^{true}, r_{i,2}^{true}, \ldots, r_{i,n}^{true} \right\} \tag{3.1}$$

where $R_{u_i}^{true}$ is user's u_i vector of "true" ratings $r_{i,j}^{true}$ over observed objects o_j. These ratings encode user's underlying, internal preferences. Pennock claims the existence of a critical distinction between true ratings and reported ratings. True ratings cannot be accessed directly by the system. Indeed the reported ratings, which users provide to the system, constitute the only accessible information. So, Pennock assumes that these ratings include Gaussian noise based on the fact that the same user may report different ratings depending on different occasions, such as his/her mood, the context of other ratings provided in the same session or on any other reason - external factor. All these factors are summarized as Gaussian noise. Statistically speaking, it is assumed that a user's u_i actual rating $r_{i,j}$ over an object o_j is drawn from an

independent normal distribution with mean $r_{i,j}^{true}$, where $r_{i,j}^{true}$ represents user's u_i true rating for object o_j. Specifically:

$$\Pr\left(r_{i,j} = x \mid r_{i,j}^{true} = y\right) \propto e^{-(x-y)^2/2\sigma^2} \tag{3.2}$$

where x is the reported rating value, y is the true rating value, that is if the reported value was *noise-free*, and σ is a *free parameter*. Given the user's personality type, his or her ratings are assumed independent. (If rating is unknown, i.e. $y = \varnothing$ (unknown), in Equation (3.2) we assign a uniform distribution over ratings.)

It is further assumed that the distribution of ratings vectors (personality types), available in the ratings matrix of the database, is representative of the distribution of personalities in the target population of users. Based on the latter, the prior probability $\Pr(R_{u_a}^{true} = \kappa)$ that the active user rates items according to a vector κ, is given by the frequency that other users rate according to κ. So, instead of explicitly counting occurrences, a random variable $R_{u_a}^{true}$ is defined which takes one out of m possible values — $R_{u_1}^{true}, R_{u_2}^{true}, \ldots, R_{u_m}^{true}$ — each one with equal probability $\frac{1}{m}$:

$$\Pr\left(R_{u_a}^{true} = R_{u_i}^{true}\right) = \frac{1}{m} \tag{3.3}$$

where $R_{u_i}^{true}$ refers to the corresponding rating vector of user u_i, $i = 1, 2, \ldots m$.

Using both previous Equations (3.2), (3.3) and given the ratings of the active user u_a, we can apply Bayes' rule to calculate the probability a user u_a is of the same personality type as any other user u_i, where $i \neq a$:

$$\begin{aligned} &\Pr\left(R_{u_a}^{true} = R_{u_i}^{true} \mid r_{a,1} = x_1, \ldots, r_{a,n} = x_n\right) \\ &\propto \Pr\left(r_{a,1} = x_1 \mid r_{a,1}^{true} = r_{i,1}\right) \ldots \Pr\left(r_{a,n} = x_n \mid r_{a,n}^{true} = r_{i,n}\right) \cdot \Pr\left(R_{u_a}^{true} = R_{u_i}^{true}\right) \end{aligned} \tag{3.4}$$

Once we have computed this quantity for each user u_i, we can find the probability distribution for active user's u_a rating for an unobserved object o_j.

$$\begin{aligned}&\Pr\left(r_{a,j}=x_j \mid r_{a,1}=x_1,\ldots,r_{a,n}=x_n\right)\\&=\sum_{i=1}^{m}\Pr\left(r_{a,j}=x_j \mid R_{u_a}^{true}=R_{u_i}^{true}\right)\cdot\Pr\left(R_{u_a}^{true}=R_{u_i}^{true} \mid r_{a,1}=x_1,\ldots,r_{a,n}=x_n\right)\end{aligned} \tag{3.5}$$

where $j \in NR = \left\{i \in N \mid r_{u,i} = \varnothing\right\}$ (set including the *not-rated* items). Finally the prediction $p_{a,j}$ for a user u_a on target item o_j is computed as follows:

$$p_{a,j} = \arg\max_{r} \Pr\left(r_{a,j}=x_j \mid r_{a,1}=x_1,\ldots,r_{a,n}=x_n\right) \tag{3.6}$$

where $r \in \left\{1,\ldots,|r|\right\}$ and $|r|$ the number of possible rating values. The algorithm has time and space complexity of order $O(mn)$, as do the memory-based methods like vector similarity and pearson correlation. This approach can be depicted as a naïve Bayesian network, as shown in Figure 3-1, having the same structure with a classical diagnostic model.

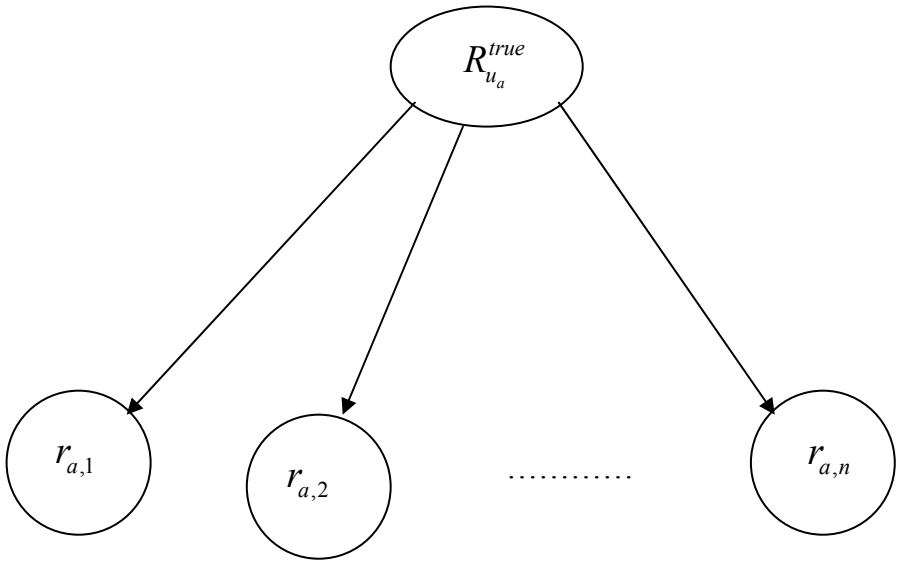


Figure 3-1: A naïve Bayesian network semantics for the PD model. Actual ratings are independent and normally distributed given the underlying "true" personality type.

According to PD, the observed ratings can be thought of as "*symptoms*" while each personality type, whose probability to be the cause we examine, as a "*disease*".

3.2.2 Features Diagnosis

If we rotate the ratings matrix by 90° we may consider the problem of recommendation formulation from another point of view introducing so the notion of *Feature Diagnosis* (FD). Based on that, for any object o_i, where $i = 1, 2 \ldots m$, and given any rating information from users available, a *type of features* which can be described as:

$$R_{o_i}^{true} = \left\{ r_{1,i}^{true}, r_{2,i}^{true}, \ldots, r_{m,i}^{true} \right\} \tag{3.7}$$

where $R_{o_i}^{true}$ is object's o_i vector of "true" ratings $r_{j,i}^{true}$ derived from users u_j. So, here, we assume that these ratings include Gaussian noise based on the fact that ratings of the same user on different items may be temporally related (i.e. if their popularities over time behave similarly). For example, during the period near St. Valentines' day romance movies may be more popular than movies about war. All these factors are summarized as Gaussian noise. These ratings encode object's underlying, internal type of features. As before, it is assumed that an object's o_i actual rating $r_{j,i}$ by a user u_j is drawn from an independent normal distribution with mean $r_{j,i}^{true}$, where $r_{j,i}^{true}$ represents object's o_i true rating by user u_j. To compute the probability of an actual rating we use Equation (3.2) again as in PD, but across all users in this case.

It is further assumed that the distribution of ratings vectors (features types), constituted by user-item matrix of the database is representative of the distribution of features in the target population of objects. Based on the latter, the prior probability $\Pr(R_{o_a}^{true} = \lambda)$ that the active object is rated according to a vector λ is given by the frequency that other objects are rated according to λ. Thus, instead of explicitly counting occurrences, a random variable $R_{o_a}^{true}$ is defined that takes one out of n possible values — $R_{o_1}^{true}, R_{o_2}^{true}, \ldots, R_{o_m}^{true}$ — each one with equal probability $\frac{1}{n}$:

$$\Pr\left(R_{o_a}^{true} = R_{o_i}^{true} \right) = \frac{1}{n} \tag{3.8}$$

where $R_{o_i}^{true}$ refers to the corresponding rating vector of object o_i, $i = 1, 2, \ldots m$.

Given the ratings of the active object o_a, we can apply Bayes' rule to calculate the probability an object o_a is of the same feature type as any other object o_i, where $i \neq a$:

$$\begin{aligned} &\Pr\left(R_{o_a}^{true} = R_{o_i}^{true} \mid r_{1,a} = x_1, \ldots, r_{m,a} = x_m\right) \\ &\propto \Pr\left(r_{1,a} = x_1 \mid r_{1,a}^{true} = r_{i,1}\right) \ldots \Pr\left(r_{m,a} = x_m \mid r_{m,a}^{true} = r_{m,i}\right) \cdot \Pr\left(R_{o_a}^{true} = R_{o_i}^{true}\right) \end{aligned} \tag{3.9}$$

Once we compute this quantity for each object o_i, we can find the probability distribution for user's u_j rating of an unobserved object o_a.

$$\begin{aligned} &\Pr\left(r_{j,a} = x_j \mid r_{1,a} = x_1, \ldots, r_{m,a} = x_m\right) \\ &= \sum_{i=1}^{n} \Pr\left(r_{j,a} = x_j \mid R_{o_a}^{true} = R_{o_i}^{true}\right) \cdot \Pr\left(R_{o_a}^{true} = R_{o_i}^{true} \mid r_{1,a} = x_1, \ldots, r_{m,a} = x_m\right) \end{aligned} \tag{3.10}$$

where $j \in NR = \left\{i \in M \mid r_{i,o} = \varnothing\right\}$ (set including the *not-rated* items). Finally the prediction $p_{j,a}$ for user u_j on the active item o_a is computed as follows:

$$p_{j,a} = \arg\max_{r} \Pr\left(r_{j,a} = x_j \mid r_{1,a} = x_1, \ldots, r_{m,a} = x_m\right) \tag{3.11}$$

where $r \in \left\{1, \ldots, |r|\right\}$ and $|r|$ the number of possible rating values. The algorithm has time and space complexity of order $O(mn)$ and it is depicted in Figure 3-2 as a naïve Bayesian network having the same structure with a classical diagnostic model.

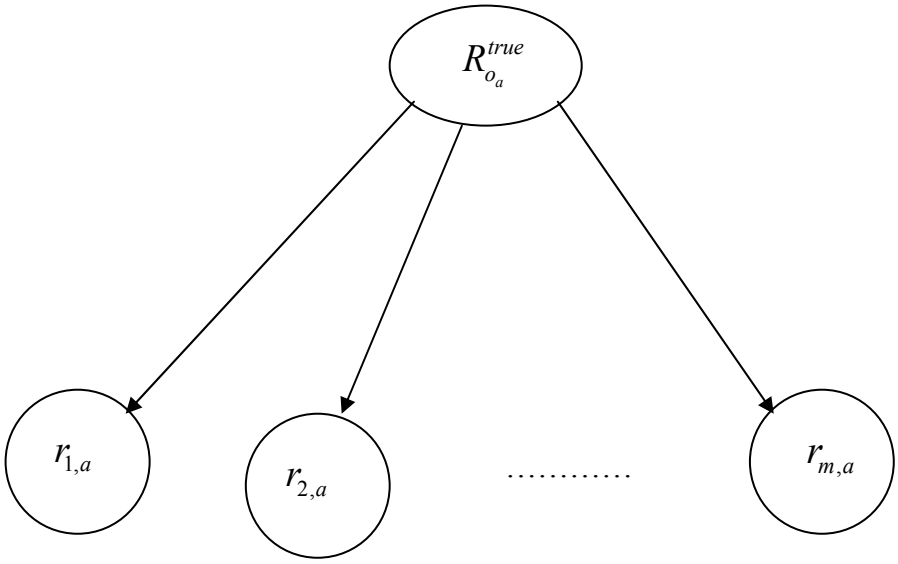


Figure 3-2: A naïve Bayesian network semantics for the FD model. Actual ratings are independent and normally distributed given the underlying "true" features type.

According to FD, the observed ratings can be thought of as "*symptoms*" while the features type as "*populations*" where symptoms may develop.

3.2.3 Context Diagnosis

The context of the objects, e.g. for a movie recommender any textual information on genres, can also provide useful information for recommendations. We assume that the probability that two objects are of the same context type, taking into account the categories in which they belong, can be derived by associating their rows in the item-category bitmap matrix. To calculate this probability we use Equation (3.12):

$$\Pr\left(C_{o_a}^{true} = C_{o_i}^{true} \mid c_1,\ldots,c_k\right) \propto \frac{\left|C_{o_a} \cap C_{o_i}\right|}{\max\left(\left|C_{o_a}\right|,\left|C_{o_i}\right|\right)} \cdot \Pr\left(C_{O_a}^{true} = C_{o_i}^{true}\right) \qquad (3.12)$$

were $C_{o_i}^{true}$ is the "true" context type of the object o_i underlying in the C_{o_i} category vector. The distribution of category vectors (context types) they belong to and is available in the category matrix of the database is assumed to be representative of the distribution of context types in the target population of objects. Based on this latter, the prior probability $\Pr(C_{o_a}^{true} = \mu)$ that the active object is classified according to a

vector μ is given by the frequency that other objects are classified according to μ. So, instead of explicitly counting occurrences, a random variable $C_{o_a}^{true}$ is defined that can have one of n values — $C_{o_1}^{true}, C_{o_2}^{true}, \ldots, C_{o_m}^{true}$ — each one with probability $\frac{1}{n}$:

$$\Pr\left(C_{o_a}^{true} = C_{o_i}^{true}\right) = \frac{1}{n} \tag{3.13}$$

Given the categories of the active object o_a, we can apply Bayes' rule to calculate the probability an object o_a is of the same context type as any other object o_i, where $i \neq a$:

Once we compute this quantity for each object o_i, we can find the probability distribution for user's u_j rating of an unobserved object o_a.

$$\begin{aligned} &\Pr\left(r_{j,a} = x_j \mid c_1, \ldots, c_k\right) \\ &= \sum_{i=1}^{n} \Pr\left(r_{j,a} = x_j \mid C_{o_a}^{true} = C_{o_i}^{true}\right) \cdot \Pr\left(C_{o_a}^{true} = C_{o_i}^{true} \mid c_1, \ldots, c_k\right) \end{aligned} \tag{3.14}$$

where $j \in NR = \left\{i \in M \mid r_{i,o} = \varnothing\right\}$ (Set including the *not-rated* items). Finally the prediction $p_{j,a}$ for user u_j on the active item o_a is computed as follows:

$$p_{j,a} = \arg\max_{r} \Pr\left(r_{j,a} = x_j \mid c_1, \ldots, c_k\right) \tag{3.15}$$

where $r \in \left\{1, \ldots, |r|\right\}$ and $|r|$ the number of possible rating values. The algorithm has time and space complexity of order $O(n)$ - considering the number k of all categories available as constant - and it is depicted in Figure 3-3 as a naïve Bayesian network having the same structure with a classical diagnostic model.

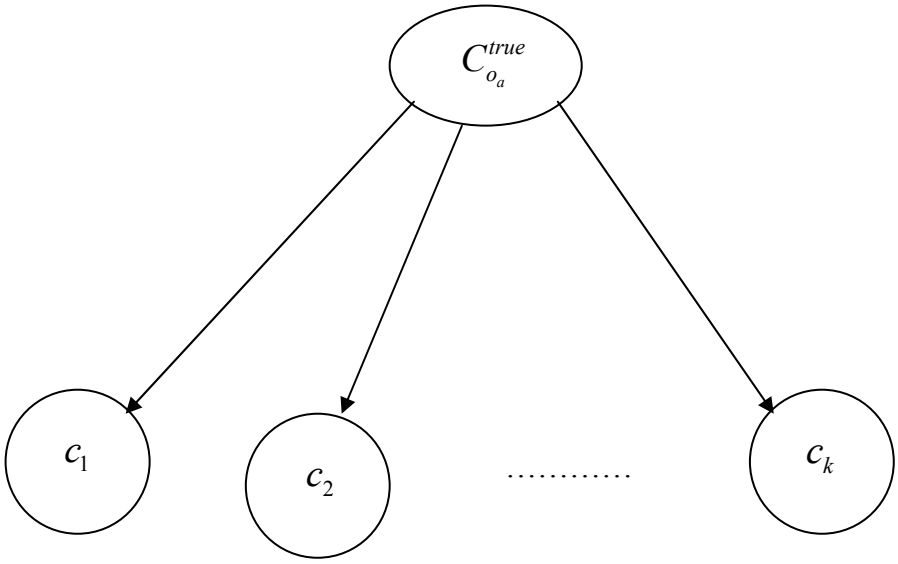


Figure 3-3: A naïve Bayesian network semantics for the CD model. Actual categories are independent and normally distributed given the underlying "true" context type.

According to CD, observed ratings can be thought of as "*symptoms*" which may develop in certain "*categories*" defined by a gamut of contextual attributes.

3.3 Combination Strategies

After describing PD and extending to a FD and a CD approach, illustrated in Figure 3-4 (a),(b),(c), we need a method to combine these techniques as shown in Figure 3-4 (d).

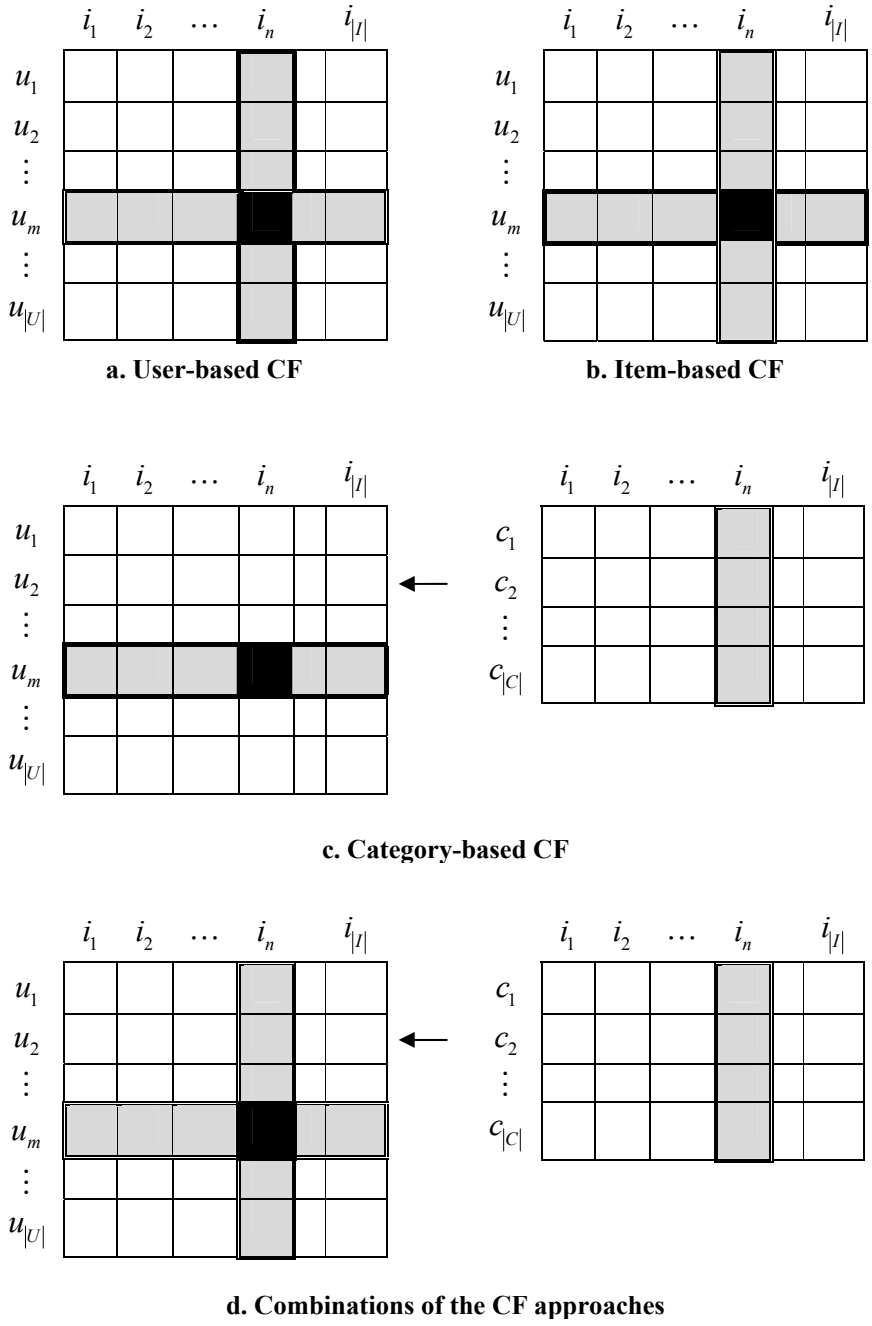


Figure 3-4: The three approaches and their combination

In the following sections we propose two combination techniques. The one is based on the product rule and the other is the sum based. Our approach is purely probabilistic and we argue that this is the major advantage.

3.3.1 Product Rule

According to the product rule, we assume that the three types of information used to make predictions are independent. We apply Bayes' rule assuming that the probability of a rating value to be the predicted value is conditional on the ratings of the user, the ratings of the object and the categories that the objects belongs to. Thus, we have:

$$\Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true}\right) = \frac{\Pr\left(R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true} \mid r_{i,j}\right)\Pr\left(r_{i,j}\right)}{\Pr\left(R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true}\right)} \tag{3.16}$$

where $\Pr\left(R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true}\right)$ is the unconditional measurement of the joint probability density. This can be expressed in terms of conditional measurement distributions as:

$$\Pr\left(R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true}\right) = \sum\sum \Pr\left(R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true} \mid r_{i,j}\right)\Pr\left(r_{i,j}\right) \tag{3.17}$$

and therefore we can neglect the denominator and focus only on the terms of the numerator which yields the following equation:

$$\Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true}\right) \propto \Pr\left(R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true} \mid r_{i,j}\right)\Pr\left(r_{i,j}\right) \tag{3.18}$$

The term $\Pr\left(R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true} \mid r_{i,j}\right)$ represents the conditional joint probability measurement distribution extracted by all the "true" vectors. At this point we assume that these vectors are conditionally independent. In order to verify whether the conditional independence of this assumption holds, we will adopt the following equation:

$$\Pr\left(R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true} \mid r_{i,j}\right) = \Pr\left(R_{u_i}^{true} \mid r_{i,j}\right) \cdot \Pr\left(R_{o_j}^{true} \mid r_{i,j}\right) \cdot \Pr\left(C_{o_i}^{true} \mid r_{i,j}\right) \tag{3.19}$$

By applying Bayes' rule to each one of the factors of Equation (3.19) we obtain the following equations:

$$\Pr\left(R_{u_i}^{true} \mid r_{i,j}\right) = \frac{\Pr\left(r_{i,j} \mid R_{u_i}^{true}\right) \cdot \Pr\left(r_{i,j}\right)}{\Pr\left(R_{u_i}^{true}\right)} \tag{3.20}$$

$$\Pr\left(R_{o_j}^{true} \mid r_{i,j}\right) = \frac{\Pr\left(r_{i,j} \mid R_{o_j}^{true}\right) \cdot \Pr\left(r_{i,j}\right)}{\Pr\left(R_{o_j}^{true}\right)} \tag{3.21}$$

$$\Pr\left(C_{o_j}^{true} \mid r_{i,j}\right) = \frac{\Pr\left(r_{i,j} \mid C_{o_j}^{true}\right) \cdot \Pr\left(r_{i,j}\right)}{\Pr\left(C_{o_j}^{true}\right)} \tag{3.22}$$

Then by substituting with Equations (3.20)-(3.22) we obtain the probability of one rating value conditional on the user's ratings, item's ratings and the category the item belongs as follows:

$$\Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true}\right) \propto \Pr\left(r_{i,j} \mid R_{u_i}^{true}\right) \cdot \Pr\left(r_{i,j} \mid R_{o_j}^{true}\right) \cdot \Pr\left(r_{i,j} \mid C_{o_j}^{true}\right) \tag{3.23}$$

Each factor in Equation (3.23) needs to be replaced by the corresponding Equations ((3.5), (3.10), (3.14)). The argument that maximizes this expression indicates the rating that user u_i is most likely to assign to object o_j:

$$\begin{aligned} p_{i,j} &= \arg\max_r \Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true}\right) \\ &= \arg\max_r \left(\Pr\left(r_{i,j} \mid R_{u_i}^{true}\right) \cdot \Pr\left(r_{i,j} \mid R_{o_j}^{true}\right) \cdot \Pr\left(r_{i,j} \mid C_{o_j}^{true}\right)\right) \end{aligned} \tag{3.24}$$

This approach is visualized in Figure 3-5 as a Bayesian network, where variables are represented as nodes and arcs represent conditional dependencies in the model described so far.

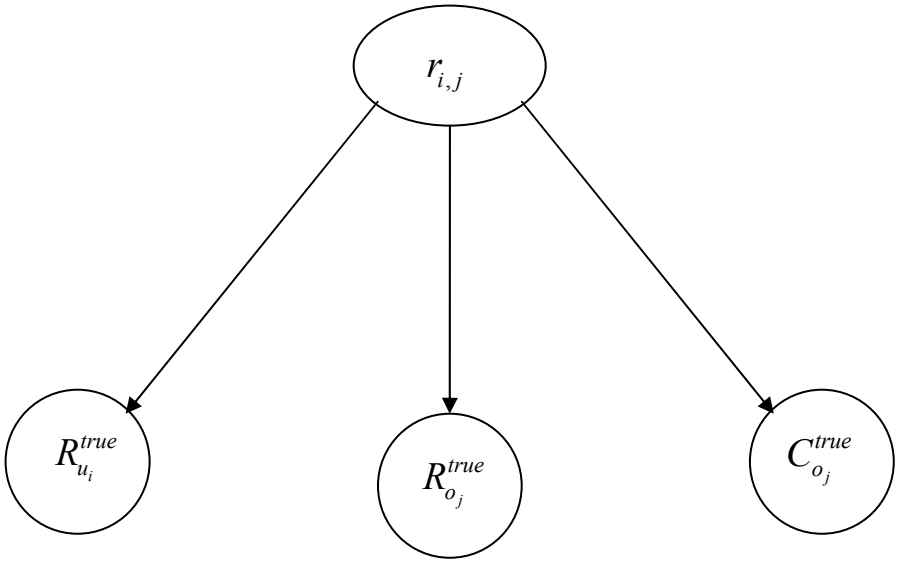


Figure 3-5: The product rule approach as a Bayesian network.

As Figure 3-5 shows, the expected rating value depends on the "true" vectors of user's personality and object's features and context type. The independence among vector types is obvious.

3.3.2 Weighted Sum Rule

To combine information from both dimensions of the user-item matrix we will make use of the Bayesian rule introducing a binary variable B that refers to the relative influence of both PD and FD methods. When variable B has value equal to 1 then the prediction comes only from user's rating vector while value 0 indicates full dependency on the object's rating vector. Under these assumptions, the conditional probability can be computed by marginalization of the binary variable B. Therefore, the equation that computes the probability distribution of objects o_j rating by user u_i is transformed to:

$$\Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}\right) = \sum_{B} \Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}, B\right) \Pr\left(B \mid R_{u_i}^{true}, R_{o_j}^{true}\right) \quad (3.25)$$

and since variable B takes only two values then we have:

$$\Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}\right) = \Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}, B=1\right)\Pr\left(B=1 \mid R_{u_i}^{true}, R_{o_j}^{true}\right) + \Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}, B=0\right)\Pr\left(B=0 \mid R_{u_i}^{true}, R_{o_j}^{true}\right) \tag{3.26}$$

By definition $r_{i,j}$ is independent from user's ratings when $B=0$ so $\Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}, B=0\right) = \Pr\left(r_{i,j} \mid R_{o_j}^{true}\right)$. The opposite stands for $B=1$; that is $\Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}, B=1\right) = \Pr\left(r_{i,j} \mid R_{u_i}^{true}\right)$. If we use a parameter ϑ to denote the probability $\Pr\left(B=1 \mid R_{u_i}^{true}, R_{o_j}^{true}\right)$ we have:

$$\Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}\right) = \Pr\left(r_{i,j} \mid R_{u_i}^{true}\right)\vartheta + \Pr\left(r_{i,j} \mid R_{o_j}^{true}\right)(1-\vartheta) \tag{3.27}$$

Next we have to extend the model to depend upon the categories that the objects belong to. To this end we introduce another binary variable B_2 that refers to the importance of predictions coming from this content-based information about the object. For $B_2=1$ predictions arrive only from rating vectors of both user and object while when $B_2=0$ specifies that the unknown rating comes only from the contextual information about the object. So, by marginalizing on the binary variable we obtain the following:

$$\Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true}\right) = \sum_{B_2} \Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true} C_{o_j}^{true}, B_2\right) \cdot \Pr\left(B_2 \mid R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true}\right) \tag{3.28}$$

and since variable B_2 can take two values only, we have:

$$\Pr\left(r_{i,j} \mid R_{U_i}^{true}, R_{O_j}^{true}, C_{O_j}^{true}\right) = \Pr\left(r_{i,j} \mid R_{U_i}^{true}, R_{O_j}^{true}, C_{O_j}^{true}, B_2=1\right)\Pr\left(B_2=1 \mid R_{U_i}^{true}, R_{O_j}^{true}, C_{O_j}^{true}\right) + \Pr\left(r_{i,j} \mid R_{U_i}^{true}, R_{O_j}^{true}, C_{O_j}^{true}, B_2=0\right)\Pr\left(B_2=0 \mid R_{U_i}^{true}, R_{O_j}^{true}, C_{O_j}^{true}\right) \tag{3.29}$$

By definition, $r_{i,j}$ is independent from the contextual information of the object when $B=0$ so $\Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}, C_{o_j}^{true}, B=0\right) = \Pr\left(r_{i,j} \mid R_{u_i}^{true}, R_{o_j}^{true}\right)$. The opposite

stands when $B=1$; that is $\Pr\left(r_{i,j} \mid R_{U_i}^{true}, R_{O_j}^{true}, C_{O_j}^{true}, B=1\right) = \Pr\left(r_{i,j} \mid C_{O_j}^{true}\right)$. If we use a parameter δ as shorthand for $\Pr\left(B=1 \mid R_{U_i}^{true}, R_{O_j}^{true}, C_{O_j}^{true}\right)$ then we have:

$$\Pr\left(r_{i,j} \mid R_{U_i}^{true}, R_{O_j}^{true}, C_{O_j}^{true}\right) = \Pr\left(r_{i,j} \mid R_{U_i}^{true}, R_{O_j}^{true}\right)(1-\delta) + \Pr\left(r_{i,j} \mid C_{O_j}^{true}\right)\delta \tag{3.30}$$

By replacing $\Pr\left(r_{i,j} \mid R_{U_i}^{true}, R_{O_j}^{true}\right)$ as it is in Equation (3.27), we have:

$$\begin{aligned} &\Pr\left(r_{i,j} \mid R_{U_i}^{true}, R_{O_j}^{true}, C_{O_j}^{true}\right) \\ &= \left(\Pr\left(r_{i,j} \mid R_{U_i}^{true}\right)\vartheta + \Pr\left(r_{i,j} \mid R_{O_j}^{true}\right)(1-\vartheta)\right)(1-\delta) + \Pr\left(r_{i,j} \mid C_{O_j}^{true}\right)\delta \end{aligned} \tag{3.31}$$

Each factor in Equation (3.31) needs to be replaced by corresponding equation. The argument that leads this expression to the maximum indicates the rating that user u_i is most likely to assign to object o_j:

$$\begin{aligned} &p_{i,j} = \arg\max_r \Pr\left(r_{i,j} \mid R_{U_i}^{true}, R_{O_j}^{true}, C_{O_j}^{true}\right) = \\ &\arg\max_r \left(\left(\Pr\left(r_{i,j} \mid R_{U_i}^{true}\right)\vartheta + \Pr\left(r_{i,j} \mid R_{O_j}^{true}\right)(1-\vartheta)\right)(1-\delta) + \Pr\left(r_{i,j} \mid C_{O_j}^{true}\right)\delta\right) \end{aligned} \tag{3.32}$$

Notice that each conditional probability factor needs to be normalized before any use of weighted sum. This approach is visualized in Figure 3-6 as a Bayesian network where variables are represented as nodes and arcs represent conditional dependencies in the model described.

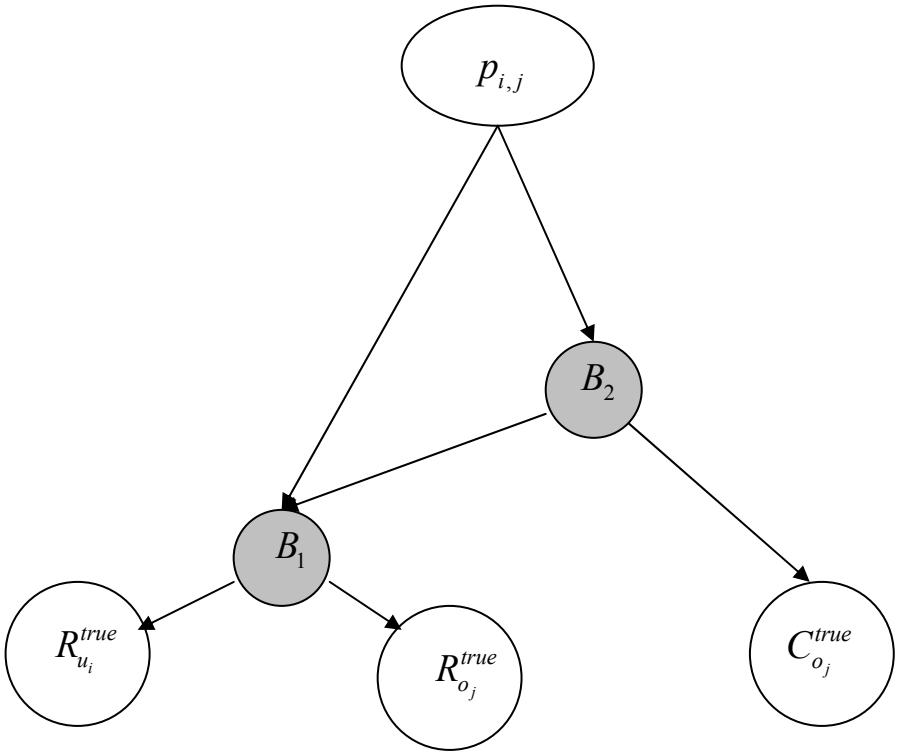


Figure 3-6: The sum rule approach as a Bayesian network.

In Figure 3-6 the clear nodes represent the independent variables - the "true" vectors - while shaded nodes represent binary values which tune between "true" vectors. It's obvious that the predicted ratings depends upon the tuning of the independent values only through their tuning and not directly.

Naïve numerical combination

So far in most systems, the numerical results obtained by each prediction technique which operated individually were combined to a single prediction. In this case, Equation (3.32) is transformed to:

$$p_{i,j} = \left(\arg\max_r \left(\Pr\left(r_{i,j} \mid R_{U_i}^{true} \right) \right) \vartheta + \arg\max_r \left(\Pr\left(r_{i,j} \mid R_{O_j}^{true} \right) \right) (1-\vartheta) \right) (1-\delta) + \arg\max_r \left(\Pr\left(r_{i,j} \mid C_{O_j}^{true} \right) \right) \delta \tag{3.33}$$

3.4 Experimental Evaluation

We conducted a set of experiments to test the efficacy of our combinations schemes in terms of sparsity. We also compare our approach with the state-of-art user-based and item-based PCC CF.

3.4.1 Dataset

We carried out our experiments using the MovieLens dataset, taken from a research recommendation site being maintained by the GroupLens project [110]. The main activity in this project is rating movies so that users can receive personalized recommendations on movies.

The MovieLens dataset contains 100.000 ratings, scaling from 0 to 5, derived from 943 users on 1682 movie titles (items) where each user has rated at least 20 movies. In our test we used the 5 training/test splits (with *training set* containing 20.000 ratings and *test set* containing 80.000) in the zip file from MovieLens and averaged the results. The ratings in the test set are used to test the accuracy of the predictions based upon data in the training set. Furthermore, in our experiments, each user's ratings in the training set are split into a set of *observed* ratings and a set of *held-out* ratings. Finally, the ratings of the observed set are the input for the prediction making procedure. For our experiments we used MATLAB 7.0 [111].

3.4.2 Set up

We tested over sparsity using the following configurations:

- User sparsity: We vary each user's number of items rated in the observed set, e.g. 5, 10, 20 ratings per user.
- Item sparsity: We vary each item's number of raters rated in the observed set, e.g. having less that 5, 10 or 20 ratings per item.
- Overall training set sparsity: Select randomly a part of the training data, e.g. 20%, 40%, 60% of the whole training set.

3.4.3 Metrics

Metrics can be classified into accuracy metrics and decision-support metrics. Accuracy metrics have been introduced to judge the accuracy of predictions, i.e., how much the predictions of items deviate from user's actual ratings. Decision-support metrics aim to evaluate how much the predictions help users to select high quality items from a set supposing binary preferences (whether user likes them or not).

3.4.3.1 Predictive Accuracy Metrics

Predictive accuracy metrics calculate how close predicted ratings have come to true user ratings. Most widely used [14], [36], [38], [92] method to represent the statistical accuracy of predictions $p_{u,i}$ for a set of R ratings $r_{u,i}$ on products is the mean absolute error (MAE):

$$MAE = \frac{\sum_{u,i} \left| p_{u,i} - r_{u,i} \right|}{R} \tag{3.34}$$

The lower the MAE value is the higher the performance indicated.

3.4.3.2 Decision-Support Metrics

Decision-support metrics have been used in information retrieval area for many years. These metrics do not take under consideration predictions or their deviations from actual ratings but rather estimate relativity between a set of ranked recommendations and the active user.

Precision, Recall, and F1

Precision and recall have been very popular in estimating information retrieval systems. Using recommender system parlance and regarding the evaluation of the recommendations, precision and recall can be defined as follows.

[85] presents an adapted variant of recall as the percentage of test set items T occurring in the recommendation list P w.r.t. the overall number of test set items $|T|$:

$$recall = 100 \cdot \frac{|T \cap P|}{|T|} \tag{3.35}$$

In other words, precision is the fraction of the recommended items that are relevant. Accordingly, precision represents the percentage of test set items T occurring in P w.r.t the overall number of items in the recommendation list $|P|$:

$$precision = 100 \cdot \frac{|T \cap P|}{|P|} \tag{3.36}$$

Recall is the fraction of the relevant items that are recommended. Another popular metric used extensively in information retrieval and recommender systems research [85][45] is the standard F1 metric which combines precision and recall in one single metric. In F1 both precision and recall have equal weight:

$$F1 = \frac{2 \cdot recall \cdot precision}{recall + precision} \tag{3.37}$$

Since the notion of relevancy is involved in these metrics, it is important to define what which items are relevant to a user. In particular, for the dataset we use, we consider the target user's relevant items known to us as the ones rated with 4 or above.

3.4.3.3 Beyond Accuracy

Despite their importance, accuracy metrics are unable to capture some traits of user's satisfaction. Still, non-accuracy metrics have largely been denied major research interest so far and have only been treated as marginally important supplements for accuracy metrics.

Coverage

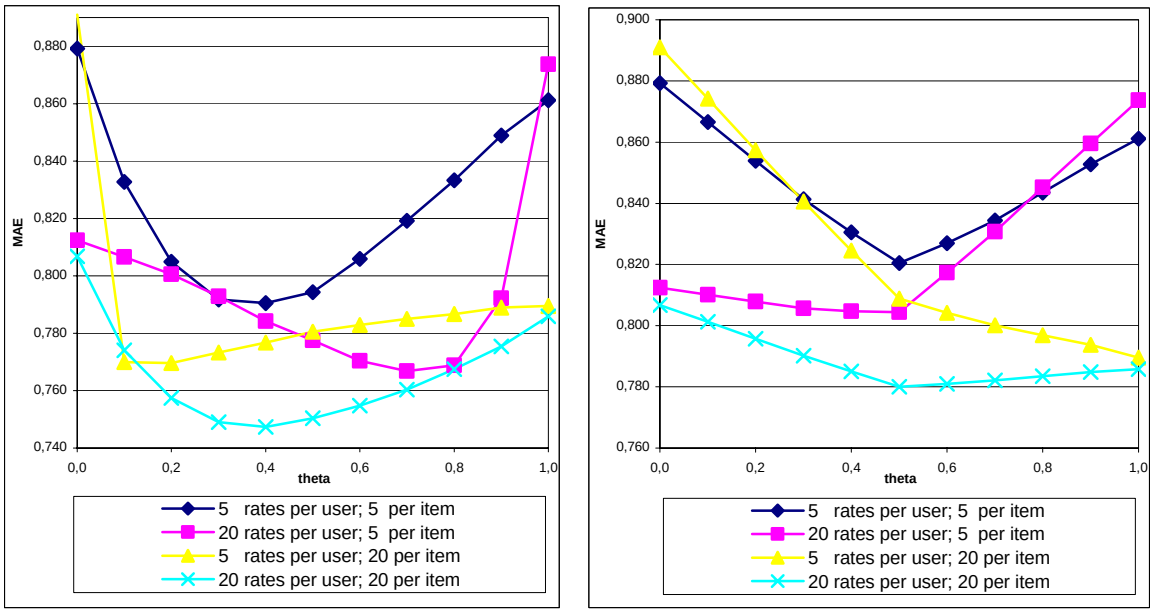
Among most non-accuracy evaluation metrics, coverage is the most frequently used [36], [38], [60]. Coverage computes the percentage of items for which predictions can be made.

3.4.4 Impact of combining multiple techniques

One could set the following question about combining several prediction techniques together into one: "Is it possible one of the techniques to degrade the final result if combined". Before proceeding to the evaluation of the two combination schemes we carried out some experiments to verify the usefulness of each technique towards the improvement of the recommendation quality.

Hence, we need to tune the weighting parameters θ and δ which exist in the sum-rule and the naïve combination scheme. Parameter θ adjusts the balance between PD and FD prediction techniques (henceforth PFD) while parameter δ adjusts the balance between PFD and CD (from now on PFCD).

First, we use MAE metric to examine the sensitivity over parameter θ (having δ set to zero). We vary this parameter from zero (pure FD) to one (pure PD). We test over user sparsity 5 and 20, test item sparsity less than 5 and 20. As Figure 3-7(a) shows, for the case of sum-rule scheme, the best results are achieved for values of θ between 0.3 and 0.7. In Figure 3-7(b), in the naïve case, the optimum results are obtained for values of θ greater than 0.5.

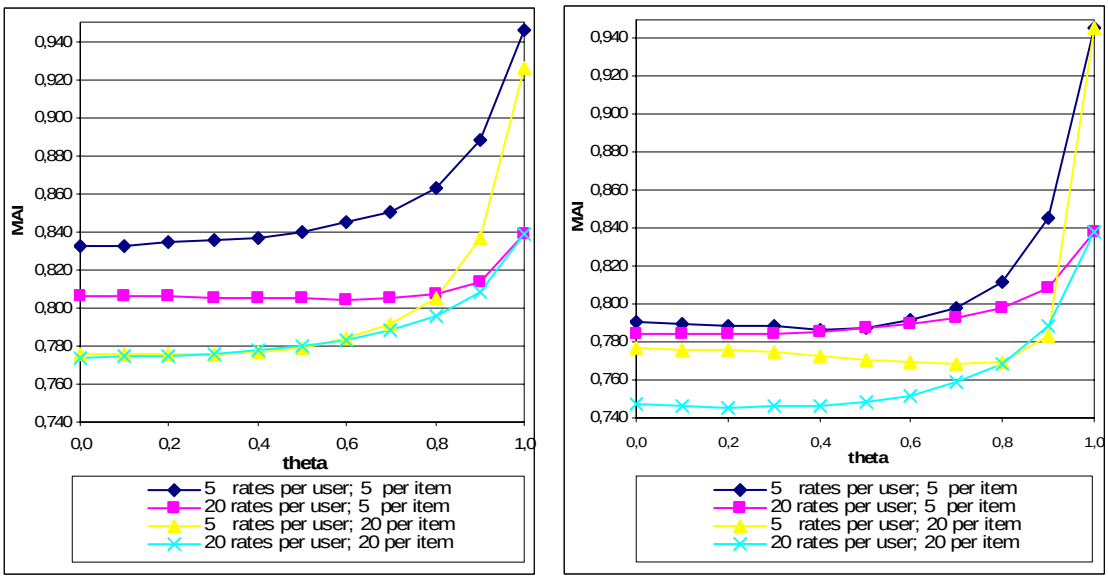


(a) sum-rule **(b) naïve combination**

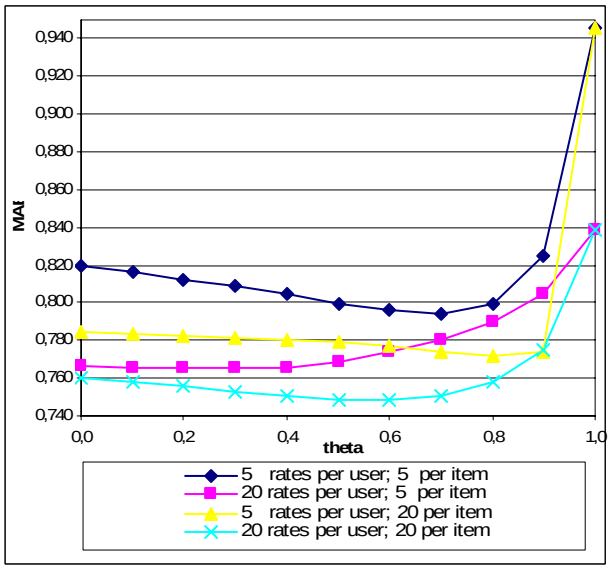
Figure 3-7: Impact of weighting parameter ϑ in sum-rule and naïve combination techniques

In Figure 3-8, for the sum-rule scheme, we use set θ to values 0.1 (henceforth *PFCDs_1)*, 0.4 (*PFCDs_2*) and 0.7 (*PFCDs_3)* and test sensitivity regarding

parameter δ. Using the same configurations over sparsity we vary δ from zero (pure rating-based approach) and one (pure content-based approach). We observe that, in case of PFCDs_1, MAE increases (lower quality) as δ increases, while PFCDs_2 and PFCDs_3 have optimum performance (lower MAE) for values of δ between 0.6 and 0.8. We set δ to be 0.1 for PFCDs_1, 0.7 for PFCDs_2 and 0.6 for PFCDs_3 and use these three configurations to examine sum-rule combination scheme over sparsity.



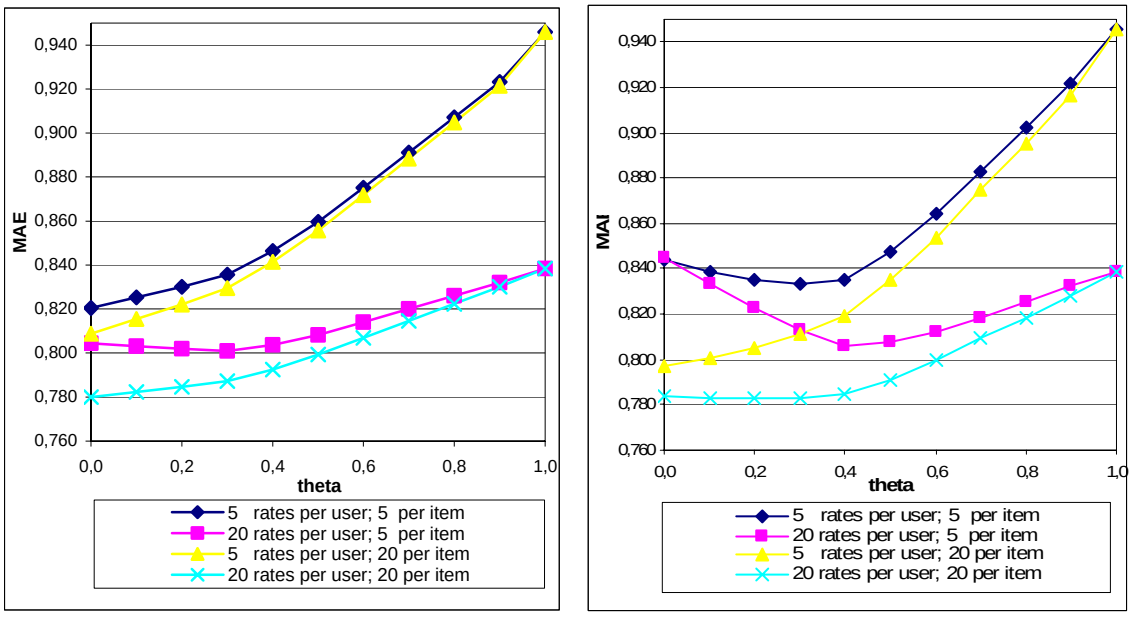
(a) $\theta = 0.1$ (b) $\theta = 0.4$



(c) $\theta = 0.7$

Figure 3-8: Impact of weighting parameter δ for different values of θ in sum-rule

In Figure 3-9, for the naïve combination scheme, we set θ to values 0.5 (henceforth *PFCDn_1*) and 0.8 (*PFCDn_2*) and test sensitivity regarding parameter δ. Using the same configurations over sparsity we vary δ from zero (pure rating-based approach) and one (pure content-based approach). We observe that, in case of PFCDn_1, MAE increases (lower quality) as δ increases, while PFCDn_2 has optimum performance (lower MAE) for values of δ lower than 0.4. We will examine the naïve combination scheme for values of δ 0.3 in PFCDn_1 and 0.4 in PFCDn_2.



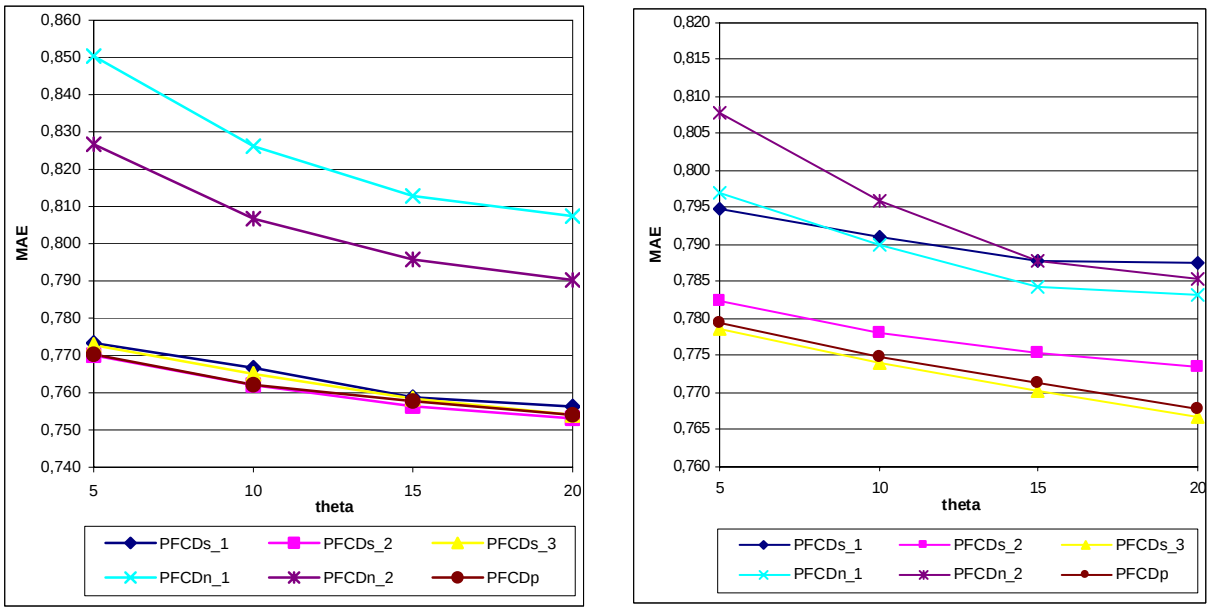
(a) **θ = 0.5** (b) **θ = 0.8**

Figure 3-9: Impact of weighting parameter δ for different values of θ in naïve combination scheme

These configurations of sum-rule and naïve combination schemes will be experimentally evaluated over sparsity along with product-rule and, later on, with state-of-art algorithms of PD, user-based and item-based PCC.

3.4.5 Overall performance

In Figure 3-10(a)(b) we compare naïve weighting with the pure probabilistic combination schemes, sum–rule and product–rule. In Figure 3-10(a) we test over user sparsity only, while in Figure 3-10(b) over item sparsity, both in terms of MAE.

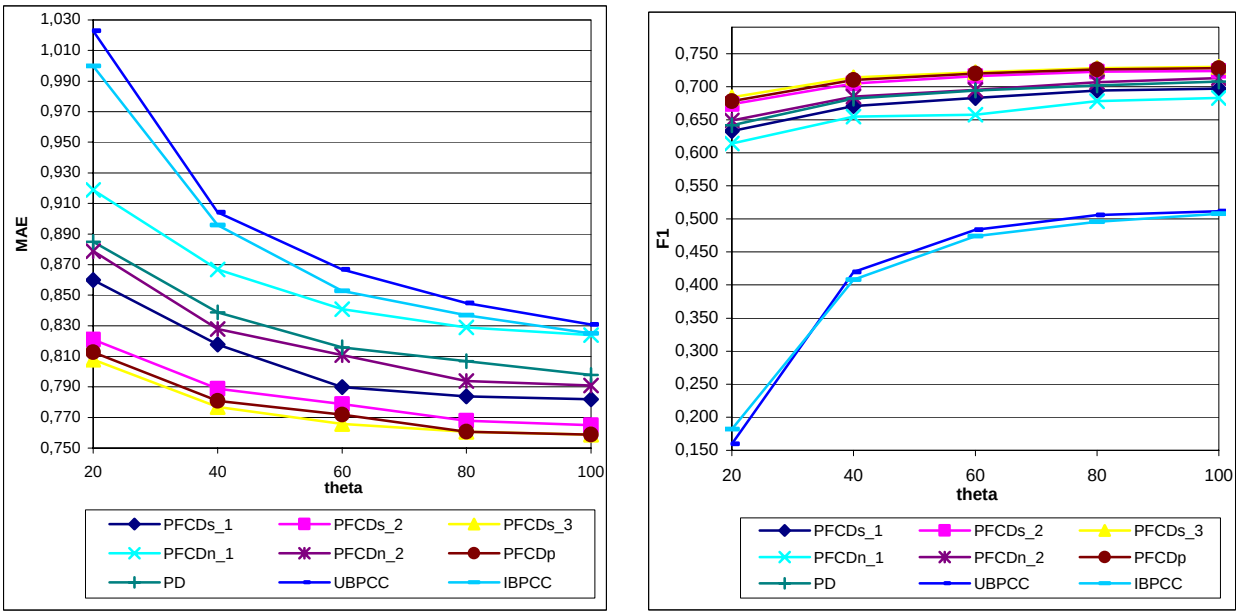


(a) User Sparsity

(b) Item Sparsity

Figure 3-10: Performance over user and item sparsity

Moreover, we compare these combinations techniques along with the state-of-the-art algorithms of collaborative filtering PD, user-based and item-based PCC. Figure 3-11 illustrates their sensitivity to overall sparsity according to MAE (Figure 3-11(a)) and F1 (Figure 3-11(b)). Table 3-3 contains the numerical results.



(a) MAE

(b) F1

Figure 3-11: Performance over different sparsity levels

	MAE					F1				
	20%	40%	60%	80%	100%	20%	40%	60%	80%	100%
PFCDs_1	0,859	0,817	0,789	0,783	0,781	0,631	0,669	0,681	0,692	0,695
PFCDs_2	0,821	0,788	0,778	0,767	0,764	0,673	0,704	0,715	0,722	0,723
PFCDs_3	0,807	0,776	0,765	0,761	0,758	0,683	0,713	0,721	0,727	0,729
PFCDn_1	0,918	0,866	0,841	0,828	0,823	0,612	0,653	0,656	0,676	0,681
PFCDn_2	0,878	0,827	0,810	0,793	0,790	0,647	0,683	0,693	0,705	0,711
PFCDp	0,812	0,780	0,771	0,760	0,758	0,677	0,709	0,719	0,725	0,727
PD	0,884	0,838	0,815	0,806	0,797	0,641	0,681	0,693	0,701	0,707
UBPCC	1,022	0,904	0,866	0,844	0,830	0,159	0,419	0,483	0,505	0,511
IBPCC	0,999	0,895	0,852	0,836	0,824	0,181	0,407	0,473	0,495	0,507

Table 3-3: Performance over different sparsity levels and comparison with other methods.

In all configurations, it is shown that the pure probabilistic schemes outperformed any of the naïve combinations. More specific, probabilistic schemes can provide statistically significant better results – up to 10% in some cases (Figure 3-11(a)).

The results differ for each pair of θ and δ. In case of naïve combinations, only PFCDn_2 improved the accuracy (measured in MAE) while the improvement in recommendation quality (measured in F1) is not quite clear. The other naïve combination, PFCDn_1, worsens results in both MAE and F1. Regarding pure probabilistic schemes, results were obviously positive in both MAE and F1 except for PFCDs_1 which shows almost the same performance as in naive weighting schemes.

3.5 Concluding remarks and discussion

In this section we proposed the use of the two basic combination schemes from theory of probabilities to overcome accuracy issues of the RS. Results have shown that a pure probabilistic combination of recommendation techniques can provide better results than naïve linear numerical weighting of results derived from each technique. However, it's worth noticing that in most cases the sum–rule (in its best configuration) has slightly outperformed the product–rule. The main reason is the sensitivity in errors which is intense in the latter case due to factorization of prediction techniques. For more details we refer to [51].

Chapter 4

Identifying Trends in Recommender Systems

4.1 Introduction

Most recommendation systems employ variations of *Collaborative Filtering* (*CF*) for formulating suggestions of items relevant to users' interests. However, CF requires expensive computations that grow polynomially with the number of users and items in the database and, thus, is not suitable for use in on-line applications. In a typical recommender system, the entire algorithm works in two separate steps. The first step is the model-building step (offline component) and the second step is the execution (online component) step.

Among all memory-based algorithms proposed, *Singular Value Decomposition* (*SVD*) has a very fast execution time. SVD is a matrix factorization technique that can produce three matrices given the rating matrix $A: SVD(A) = U \times S \times V^T$. Details of SVD can be found in [25] however, suffice it to say that the matrices U, S, and V can be reduced to construct a rank-k matrix, $X = U_k \times S_k \times V_k^T$ that is the closest approximation to the original matrix. The first model-building component is very time-consuming. Sarwar [85] and, later, Brand [13] proposed a highly scalable incremental algorithm for the execution of this component. However, in SVD-based algorithms, the predictions are not formulated by considering ratings from other but similar users or items because dimensionality reduction is applied only for purposes as factorization and synopsis of the initial user-item matrix. This may be the main reason why "SVD works well for some systems while less well for some other" [85].

To import the concept of CF, Symeonidis et al. [96][97] used SVD to produce a "thinner" matrix which was used for the estimation of similarities among users or items. Despite the substantial performance gains they used existing SVD to represent new information (folding-in) resulting in slight quality loss due to the non-orthogonality of the resultant space.

We could address the scalability problem based on incremental updates of user-to-user personality similarities as in [70]. *Incremental Collaborative Filtering* (*ICF*) is not an approximation-based method and gives the potential for high-quality recommendation formulation. However, this approach hurts memory complexity because the recommender system needs to process a huge amount of data as the number of users or items increases, as discussed in previous chapter.

4.2 Proposed Framework

In this propose framework we aim to migrate from user ratings to user trends for the computation of similarities. Information about trends based on rating activity can be distilled by tracking and mapping users into a subspace of the item space. Moreover, since all recommendation systems are non-stationary environments, we adopt an incremental PCA algorithm for dynamic tracking-diagnose of the item subspace.

Algorithm: Incremental Trend Diagnosis

- **Preprocess:** find the first p most significant eigenvectors and their corresponding eigenvalues that consist of the new item subspace (use of Principal Component Analysis). Map users into the new subspace.
- **Incremental Updating:** Each time a modification in rating matrix occurs (i.e. update/submission of rating, new user, new item) we apply the incremental PCA to track the new subspace of the items and then apply a new user mapping.
- **Prediction:** Given an active user's u_a rating profile find the probability that he is of the same personality type with the other users based on trends. Predict the rating of a particular item according to the rating behavior of other users on the item in question. Combine prediction coming from both actual ratings and filled ratings.

Figure 4-1: Description of the Incremental Trend Diagnosis framework

Figure 4-1 contains a brief description of our framework. As shown, we adopt Pennock's *Personality Diagnosis* (PD) [73] algorithm for the prediction step, therefore we name our framework "*Incremental Trend Diagnosis*" (ITD). In the following paragraphs we discuss in detail each block of our proposal.

4.2.1 Data Missing

As discussed in previous chapter, data sparsity is a fundamental problem in recommendation systems. A lot of research has been carried out to overcome these limitations.

To circumvent this limitation of the rating datasets, [85] proposed using average values in the empty cells of the rating matrix. An alternate method proposed by Srebro et al. [95] finds a model that maximizes the log-likelihood of the actual ratings by an *Expectation Maximization* (*EM*) procedure. In the E-step, missing entries of A are replaced with the values of current X creating an expected complete matrix A'. Next, during the M-step, $SVD(A)$ is performed to create an updated X. This EM process is guaranteed to converge. Upon convergence, the final X represents a linear model of the rating data, and the missing entries of the original A are filled with predicted values.

Unfortunately, all the methods above require expensive computations to fill the missing values in the rating matrix. Nevertheless, approaches less expensive than previous SVD-based methods were proposed. [59] is an example, where a naive Bayesian text classifier was used to exploit content information in the CF process. Results in [59] indicated that a content-boosted background improves the performance in CF. Moreover, by making the rating matrix denser, it is more possible to increase the variance of and decrease the variance around the most principal axes (components) when applying PCA.

In this framework, we propose the use of the *Context Diagnosis* (*CD*) algorithm presented in section 3.2.3 and which is analogous to [59]. It's a quite simple algorithm to implement and has time and space complexity of order $O(mn)$.

4.2.2 Tracking Subspace

Principal Component Analysis (*PCA*), or the subspace method, has been extensively studied in the field of computer vision and pattern recognition. One of the attractive characteristics of PCA is that a high dimensional vector can be represented by a small number of orthogonal basis vectors, i.e. the principal components.

Our approach applies PCA in the CF process. To smoothen the progress of the discussion, we will use the example illustrated in Figure 4-2 where $u_1 - u_3$ are users and $i_1 - i_4$ are items. The values in italics indicate null values (not rated) filled using the CD content-based algorithm (section 3.2.3):

	i_1	i_2	i_3	i_4
u_1	*4*	*3*	*3*	2
u_2	*3*	*4*	4	3
u_3	1	3	3	*2*

a. Training set (user-item matrix)

	i_1	i_2	i_3	i_4
i_1	2.33	0.16	0.16	0.16
i_2	0.16	0.33	0.33	0.33
i_3	0.16	0.33	0.33	0.33
i_4	0.16	0.33	0.33	0.33

b. Covariance matrix

Figure 4-2: Example of training set (a) and its covariance matrix (b). The content-based filled values of the rating matrix are shaded.

By eigen-decomposing the covariance matrix of the training set, we produce two matrices: a diagonal eigenvalues matrix and an eigenvectors matrix (Figure 4-2).

	i_1	i_2	i_3	i_4
i_1	2.33	0.16	0.16	0.16
i_2	0.16	0.33	0.33	0.33
i_3	0.16	0.33	0.33	0.33
i_4	0.16	0.33	0.33	0.33

a. Initial covariance matrix $\tilde{C}$

2.39	0	0	0
0	0.94	0	0
0	0	0.01	0
0	0	0	0

b. Eigenvalues matrix Λ_{pp}

-0.9	-0.2	0	0
-0.1	0.56	-0.4	0.7
-0.1	0.56	-0.4	-0.7
-0.1	0.56	0.81	0

c. Eigenvectors matrix U_{pp}

Figure 4-3: Example of eigen-decomposing initial covariance matrix (a) into principal eigenvalues (b) and eigenvector (c) matrices.

These matrices obtained by this method are able to give by performing multiplication, using Equation (4.1), the initial covariance matrix:

$$C = U_{mm} \times \Lambda_{mm} \times U_{mm}^{T} \tag{4.1}$$

where the columns of U_{mm} are the eigenvectors of C and the diagonal matrix Λ_{mm} is comprised of all eigenvalues of C. The covariance matrix can be approximated by preserving the p principal eigenvector and their corresponding eigenvalues, and so, reducing the $m \times m$ initial covariance matrix C to include only the p largest components according to Equation (4.2):

$$\tilde{C} = U_{mp} \times \Lambda_{pp} \times U_{mp}^{T} \approx C \tag{4.2}$$

where the columns of U_{mp} are the p principal eigenvectors of C and the diagonal matrix Λ_{pp} is comprised of p principal corresponding eigenvalues of C. Then, the reconstructed matrix is the closest rank-p approximation $\tilde{C}$ of the initial covariance matrix C, as shown in Figure 4-4.

	i_1	i_2	i_3	i_4
i_1	1.97	0.11	0.11	0.11
i_2	0.11	0.32	0.32	0.32
i_3	0.11	0.32	0.32	0.32
i_4	0.11	0.32	0.32	0.32

a. Approximation of covariance matrix $\tilde{C}$

2.39	0
0	0.94

b. Most significant eigenvalues matrix Λ_{pp}

-0.9	-0.2
-0.1	0.56
-0.1	0.56
-0.1	0.56

c. Most significant eigenvectors matrix U_{mp}

Figure 4-4: Approximation of covariance matrix (a) using principal eigenvalues (b) and eigenvector (c) matrices.

It is possible to reveal the major trends by tuning the number p of the principal components. To estimate the optimum value p we first have to determine the percentage of information we want to preserve compared to the original matrix. Therefore, a p-dimensional space is created where each of the p dimensions corresponds to a distinctive rating trend.

We first have to determine the information percentage we want to preserve compared to the original matrix in order to tune over variable p. In the running example we have created a 2-dimensional space preserving 99% of the initial information of the covariance matrix ((2.39+0.94)/(2.39+0.94+0.01)).

4.2.3 User Mapping

The next step is to map user vectors to the reduced *p*-dimensional item subspace. To this aim, given the current ratings of each user (both actual and filled ratings) we use Equation (4.3):

$$\tilde{u} = u \times U_{mp} \tag{4.3}$$

In the current example, we map users $u_1 - u_3$ into a 2-dimensional subspace, as it is shown in Figure 4-5.

	i_1	i_2	i_3	i_4
u_1	4	3	3	2
u_2	3	4	4	3
u_3	1	3	3	2

a. Users in 4-dimensional item space

2.39	0
0	0.94

b. Λ_{pp}

-0.9	-0.2
-0.1	0.56
-0.1	0.56
-0.1	0.56

c. U_{mp}

	p_1	p_2
u_1	-4.4	3.7
u_2	-3.8	5.5
u_3	-1.7	4.2

d. Users in 2-dimensional item subspace (2 trends)

Figure 4-5: Example of mapping users from item space (a) using eigenvalues (b) and eigenvectors (c) to trends subspace (d)

This $m \times p$ "*thin*" matrix is the p-dimensional representation of the m users for the n items. So far, complexity computation failed to give real-time performance and behavior of the memory-based algorithms. From now on, the quantity of the items is not considered as important factor.

4.2.4 Prediction Generation

We will use for the prediction step Pennock's PD algorithm [73]. Before we proceed to this step we need to find the probability that a user's personality is of the same type as any another user based on trends. To this end we assign to Equation (3.4) the reduced user coordinates to the item subspace obtained by Equation (4.3) (i.e. user trend vectors).

In order to predict user's i rating over an unrated item j, there are two categories of ratings to take under consideration: actual ratings and ratings estimated in earlier step. Having two different sources of ratings to handle we propose the use of the weighted sum rule to combine them. Thus, we transform previous Equation (3.27) to:

$$\Pr\left(r_{i,j} \mid AR_j, PsR_j\right) = \Pr\left(r_{i,j} \mid AR_j\right)\vartheta + \Pr\left(r_{i,j} \mid PsR_j\right)\left(1-\vartheta\right) \tag{4.4}$$

where AR_j consists of the actual ratings for item j, PsR_j contains the pseudo ratings of item j. $\Pr\left(r_{i,j} \mid AR_j\right)$ and $\Pr\left(r_{i,j} \mid PsR_j\right)$ are the probabilities that user i would assign rating $r_{i,j}$ according to his personality type and considering as known only the actual or only the pseudo ratings of the item j respectively (Equation (3.5)). The rest are treated as unknown.

4.2.5 Incremental Updating

The conventional methods of PCA, such as SVD and eigen-decomposition, perform in batch-mode with a computational complexity of $O\left(min(m,n)^3\right)$ where m is the number of the users and n the number of the items. Undoubtedly these methods are computationally expensive when dealing with large scale problems where both sizes of users and items are large. To address this problem, many researchers have been working on incremental algorithms.

It is also important to distinguish an incremental algorithm from an iterative algorithm. The former performs in the manner of prototype growing from training example *1, 2* ... to *t*, the current training example, while the latter iterates on each learning step with all the training examples *1, 2* ... and *N* until a certain stop condition

is satisfied. Therefore, for the PCA problem, the complexity of algorithms in the order from the lowest to highest is: incremental, batch-mode and iterative algorithm.

For the reasons mentioned, we adopt the incremental algorithm in [55] which is concise and easy to be implemented. In this approach the PCA model updating is performed directly from the previous eigenvectors and a new observation vector. We also extend this algorithm to take under consideration the case of the insertion of a new item. The real-time performance can be significantly improved over the traditional batch-mode algorithm.

Assume again that the covariance matrix C can be approximated by the first p significant eigenvectors and their corresponding eigenvalues as in Equation (4.2). The key idea in this incremental algorithm lies in expressing the new covariance matrix as a product of a matrix A and its inverse:

$$C^{new} = AA^T \tag{4.5}$$

Matrix A should be built using the approximation of the old covariance matrix. But, instead of the $n \times n$ matrix C^{new} we eigen-decompose a smaller matrix B, such that:

$$B = A^T A \tag{4.6}$$

yielding eigenvectors $\left\{\upsilon_i^{new}\right\}$ and eigenvalues $\left\{\lambda_i^{new}\right\}$ which satisfy

$$B\upsilon_i^{new} = \lambda_i^{new}\upsilon_i^{new}, i = 1, 2, \ldots, p+1 \tag{4.7}$$

Left multiplying by A on both left sides and using Equation (4.6) we have

$$AA^T A\upsilon_i^{new} = \lambda_i^{new} A\upsilon_i^{new} \tag{4.8}$$

Defining

$$u_i^{new} = A\upsilon_i^{new} \tag{4.9}$$

and then using Equations (4.5) and (4.9) in (4.11) leads to

$$C^{new}u_i^{new} = \lambda_i^{new}u_i^{new} \tag{4.10}$$

i.e. u_i^{new} is an eigenvector of C_i^{new} with eigenvalue λ_i^{new}. Our aim is to find a way to build matrix A for each case of training set update.

4.2.5.1 New rating

In case of a new rating, whether this is about a new submission or an update of an existing one, we have to use our content-based technique to update the missing ratings and, along with the actual ratings, to form active user's vector. Next we treat this vector as a new observation x. Note that in this context we use x to denote the mean-normalized observation vector, i.e.:

$$x = x' - \mu \tag{4.11}$$

where x' is the original vector and μ is the current mean vector. For a new x, if we assume the updating weights on the previous PCA model and the current observation are α and $1-\alpha$ respectively, the mean vector can be updated as:

$$\mu^{new} = \alpha\mu + (1-\alpha)x' = \mu + (1-\alpha)x \tag{4.12}$$

Construct $p+1$ vectors from the previous eigenvectors and the current observation vector:

$$y_i = \sqrt{\alpha\lambda_i}u_i \tag{4.13}$$

$$y_{p+1} = \sqrt{(1-\alpha)}x \tag{4.14}$$

where $\{u_i\}$ and $\{\lambda_i\}$ are the current eigenvectors and eigenvalues. The PCA updating problem can then be approximated as an eigen-decomposition problem on the $p+1$ vectors. The desired $n\times(p+1)$ matrix A can then be defined as:

$$A = \left[y_1, y_2 \ldots, y_{p+1}\right] \tag{4.15}$$

The algorithm is described in Figure 4-6. To update user's filled ratings and the mean vector we need $O(n)$ computations. The computations to construct matrices A and B cost $O(np)$ and $O(np^2)$. The time complexity to eigen-decompose matrix B is reported to be $O(p^3)$ in [55]. The new eigenvectors $\{u_i^{new}\}$ are computed in $O(np)$ steps. Assuming that the number of features, i.e. dimensions of item subspace, is a constant, the cost of the incremental algorithm becomes $O(n)$.

The incremental algorithm of PCA for rating update

Construct the initial PCA from the first $q\,(q \geq p)$ observations.

for all new observation x **do:**

Update the mean vector *(4.12)*;

Compute $y_1, y_2, \ldots, y_p$ from previous PCA *(4.13)*;

Compute y_{p+1} *(4.14)*;

Construct matrix A *(4.15)*;

Compute matrix B *(4.6)*;

Eigen-decompose B to obtain eigenvectors s $\{\upsilon_i^{new}\}$ and eigenvalues $\{\lambda_i^{new}\}$;

Compute new eigenvectors $\{u_i^{new}\}$ and re-normalize *(4.9)*;

end for

Figure 4-6: Synopsis of the incremental PCA algorithm for the case of rating update

About the incremental PCA described in Figure 4-6 it is worth noticing that:

- The actual computation for matrix B only occurs for the elements of the $(p+1)^{th}$ row or the $(p+1)^{th}$ column since $\{u_i\}$ are orthogonal unit vectors, i.e. only the elements on the diagonal and the last row/column of B have non-zero values.

- The update rate α determines the weights on the previous information and new information. Like most incremental algorithms, it is application-dependent and has to be chosen experimentally. Also, using this updating scheme, the old information stored in the model decays exponentially over

time revealing mostly new trends. That is the meaning of using the term "trend" to characterize this approach.

4.2.5.2 New user insertion

The case of updating can be handled as the previous. We consider the new user, i.e. his rating vector, as a new observation and follow the same steps in Figure 4-6.

4.2.5.3 New item insertion

When a new item is inserted in the database first we need to predict all ratings for each user according to the content-based technique. From this step we will obtain a new column in the rating matrix R (as in Figure 4-2). We use c to denote the new item column.

In this update case, the covariance matrix grows by one row/column. To this aim it is first necessary to calculate the covariance of the new item with the already existing ones.

$$\beta = c^T \times R \tag{4.16}$$

Again, as before, construct p vectors from the previous eigenvectors:

$$y_i = \sqrt{\lambda_i} u_i \tag{4.17}$$

where $\{u_i\}$ and $\{\lambda_i\}$ are the current eigenvectors and eigenvalues. Next, we construct the $n \times p$ matrix A:

$$A = \left[y_1, y_2 \ldots, y_p \right] \tag{4.18}$$

Since the covariance matrix grows by one dimension, the same happens to the eigenvectors. Let's consider a matrix X containing the coordinate of each eigenvector for the new dimension, such as:

$$X = \left[x_1, x_2 \ldots, x_p \right] \tag{4.19}$$

where x_i is the new coordinate for the i component. X is related to the new item. By multiplying one row of A, which corresponds to a particular item, with X we get their covariance. Since the covariance of the new item and the rest items is already known by Equation (4.16), to find X, we need to solve the n equations for p unknown values. It's worth noticing that the eigenvectors, as an orthogonal basis, are linearly independent which allows us to use the pseudoinverse of A to solve this system. Thus:

$$\begin{aligned} A^{+} \times A \times x^{T} &= A^{+} \times \beta \\ I \times x^{T} &= A^{+} \times \beta \\ x^{T} &= (A^{T} \times A)^{-1} \times A^{T} \times \beta \end{aligned} \tag{4.20}$$

The PCA updating problem can then be translated as an eigen-decomposition problem on the p vectors. The desired $(n+1) \times p$ new matrix A can then be defined as:

$$A = \begin{bmatrix} A \\ X \end{bmatrix} \tag{4.21}$$

The incremental algorithm of PCA for new item insertion

Construct the initial PCA from the first $q\,(q \geq p)$ observations.

for all new observation x **do**

Compute item ratings using the CD algorithm;

Calculate the new item covariance *(4.16)*;

Compute $y_1, y_2, \ldots, y_p$ from previous PCA *(4.17)* and construct A *(4.18)*;

Solve system *(4.20)* to find x;

Compute new matrix A *(4.21)*;

Compute matrix B *(4.6)*;

Eigen-decompose B to obtain eigenvectors $\{\upsilon_i^{new}\}$ and eigenvalues $\{\lambda_i^{new}\}$;

Compute new eigenvectors $\{u_i^{new}\}$ and re-normalize*(4.9)*;

end for

Figure 4-7: Synopsis of the incremental PCA algorithm for the case of new item insertion

The algorithm is described in Figure 4-7. To fill all missing new item ratings we need computations of order $O(mn)$. The calculation of new item covariance has

complexity $O(mn)$ as well. Matrices A and B require $O(np^3)$ and $O(np^2)$ to be built. The time complexity to eigen-decompose matrix B is of order $O(p^3)$. The new eigenvectors $\{u_i^{new}\}$ are computed in $O(np)$ steps. Considering the number of principal components as constant, the cost of the incremental algorithm in the case of new item insertion becomes $O(mn)$.

4.3 Complexity Issues

In this section, we discuss the computational complexity of the CF in the following forms: naive CF, ICF as in [70] and ITD. We present the worst cases of each. Our study spans in two directions, the one refers to the maintenance of any essential update information and the other refers to information retrieval regarding similarities among users. For each direction we report both time and space complexities. In our analysis we do not consider the time and memory cost on the rating matrix since it is applied in all forms of CF.

In case of naive CF, there's no further information to maintain than updating user-item matrix which costs $O(1)$ operations. In order to calculate how similar active user is to all others $O(mn)$ are needed.

In the case of ICF algorithm user-to-user similarities are computed incrementally at the time of rating activity and not at the time that a recommendation is requested. The complexity of this operation is $O(mn)$, as $m-1$ similarities need to be updated and at most n items need to be examined for each user. Since user similarities are considered pre-computed, the retrieval cost is of the order of $O(1)$ as only one operation (only one query) is enough to get this information (user similarities) from where it is stored.

In the case of ITD algorithm first we need to update item subspace. In the worst case, i.e. case of new item insertion, this procedure costs $O(mn)$. When subspace is updated, it's necessary to map users to the new coordinates. The complexity of this operations is $O(pmn)$. Total maintenance cost is approximated to $O(pmn)$. Since

new user coordinates are limited to p the cost to compute user-to-user similarities using the "thin" user-trends table is the of order $O(pm)$.

	Classic CF	Incremental CF	Incremental TD
For information maintenance	$O(1)$	$O(mn)$	$O(pmn)$
For similarity retrieval	$O(mn)$	$O(1)$	$O(pm)$

Table 4-1: Time complexity table for Classic CF, ICF and ITD

Time complexities are summarized in Table 4-1. In order to deal with the most expensive tasks, major e-commerce systems prefer to carry out expensive computations offline and feed the database with updated information periodically [56]. In this way, they succeed to provide quick recommendations to users, based on pre-computed similarities. These recommendations however, are not of the highest accuracy, because ratings submitted between two offline computations are not considered. Thus, the offline computation method may be detrimental to new or obscure users and items due to their almost undeveloped profile.

Regarding space complexities, in naïve CF there is no additional storage requirements besides the rating matrix. However, in case of ICF all user-to-user similarities ($\frac{m\times(m-1)}{2}$ user pairs) need to be stored in a new matrix yielding extra memory cost of the order of $O(m^2)$. Finally, for ITD we need to store user-trends $p\times n$ matrix and a $p\times m$ matrix containing the item-space eigenvectors. Table 4-2 contains space complexities of all forms of CF.

	Classic CF	Incremental CF	Incremental TD
For information maintenance	-	$O(m^2)$	$O(p(m+n))$
For similarity retrieval	-	$O(m^2)$	$O(pm)$

Table 4-2: Space complexity table for Classic CF, ICF and ITD.

In the case of ITD, parameter p (number of principal components) is constant and may be neglected. Any use of this parameter above is for purposes/reasons that we will discuss later (Section 4.4.2). As arises from our cost analysis, ITD has less cost than both naïve and incremental CF.

4.4 Experimental evaluation

We conducted a set of experiments to test the effectiveness of our framework and also against other state-of-art algorithms such as Pearson correlation coefficient collaborative filtering and vector similarity in both user-based and item-based approaches. We also test the accuracy of our predictions using the proposed incremental approach.

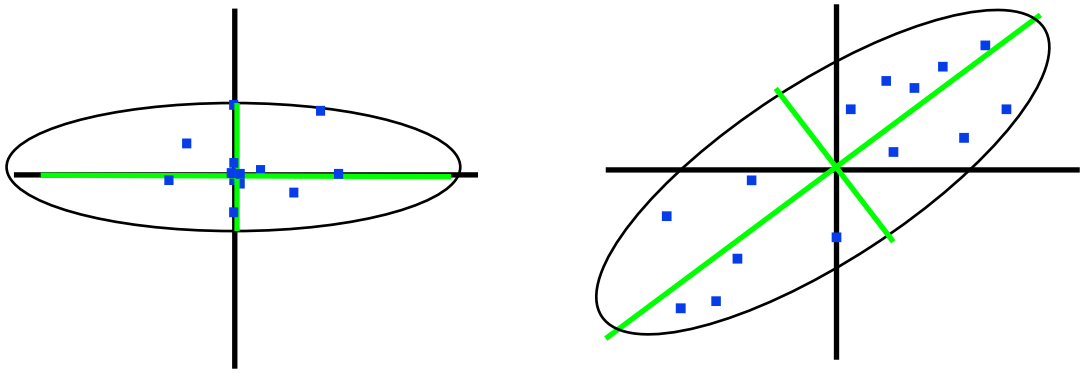
4.4.1 Set up

The dataset we used in our experiments is again the MovieLens [110] dataset which contains 100.000 ratings, scaling from 0 to 5, derived from 943 users on 1682 movie titles (items). In our tests we split our dataset into a *training set* and a *test set*. The ratings in the test set are used to test the accuracy of the prediction based upon data in the training set. We repeated our experiments using the 5 training/test splits in the zip file from MovieLens and averaged the results. For our experiments we used MATLAB 7.0 [111].

We are concerned in evaluating prediction accuracy of the algorithm compared to others and in incremental updating. The metric used are Mean Absolute Error (MAE) while also F1 is considered important as well. Descriptions of these metrics are contained in previous chapter.

4.4.2 Results

PCA is commonly used for sparse matrices. One essential question could be the following: “Why bother filling the missing values of the rating matrix and making it dense?”



(a) sparse matrix (b) dense matrix

Figure 4-8: Principal axes in a two-dimensional item space

Two answers could be given to this question. First, according to [14][102] content-boosted algorithms provide better results. Second, by using a content-predictor to fill the missing ratings, we aim to enforce high variance over sparsely rated items and thus, affecting the formation of the principal axes so that more information about the relation between the items is included. This could lead to a system similar to Figure 4-8 (b).

In Figure 4-8 a simple example in two dimensions (items) is illustrated. The first system Figure 4-8 comes from a sparse matrix where due to plenty of zeros the observations (users) are gathered close to the centroid (mean of all directions-items). The principal axes in this case cannot provide any information about the relationship between the two items. What information about the relationship among users could we obtain from an item which is rated all the time with the same value? The second system, Figure 4-8, is of a dense matrix. As is obvious, more information about the two items can be obtained by preserving the most principal components of that system.

We run PCA in both sparse and dense matrices and test the amount of information preserved regarding the number of principal components. Results in Figure 4-9 confirm our initial assumption. For example, by keeping 20 principal components we preserve almost 35% of information of the covariance matrix when operate on a sparse matrix while in case of a dense matrix we keep 85% of the information respectively. On the other hand in order to preserve 80% of the initial info we should store more than 100 principal components in case of a sparse matrix when only the 10

principal components are enough in the other case of a dense matrix. This has a tremendous effect on operational cost, as we have proved above that time and space complexities are dependent on the number of principal components in our proposed framework. Thus, recall to our example, in the case of a sparse matrix the cost is one order of magnitude greater than in the case of a dense matrix.

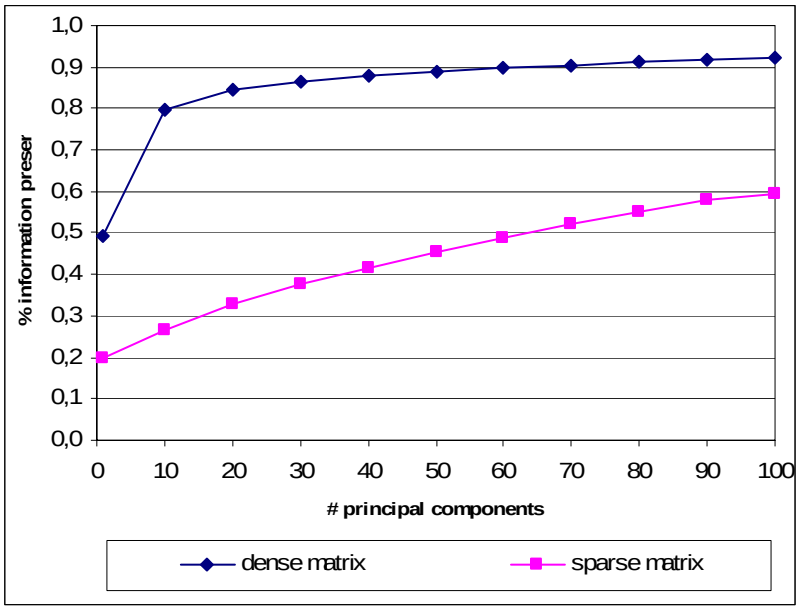


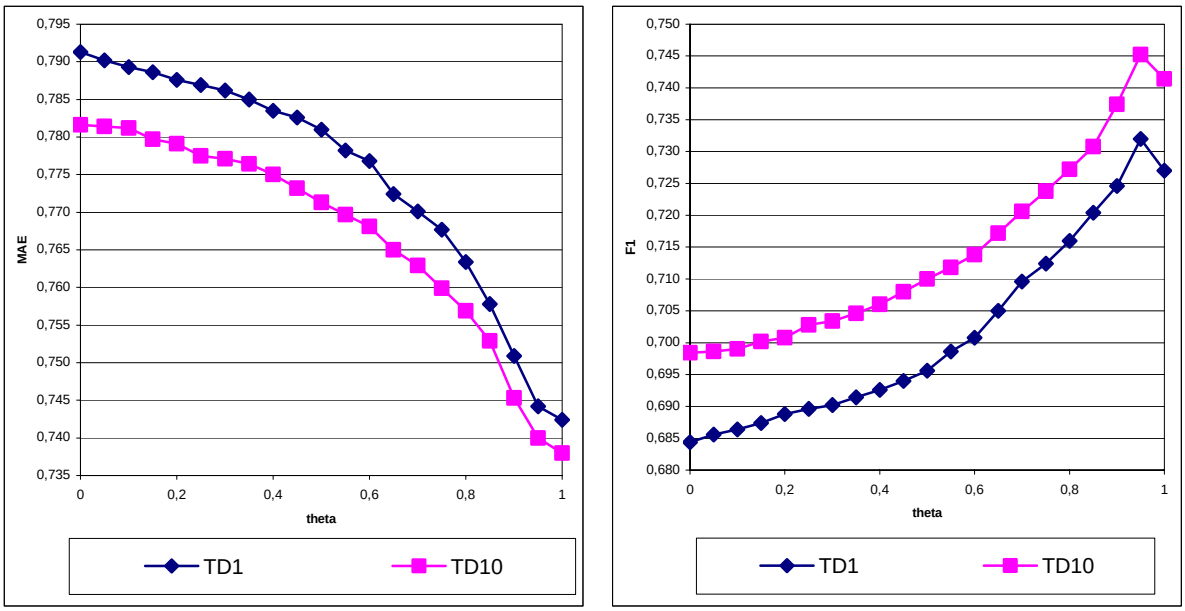
Figure 4-9: Information preserved over number of principal components

In the following experiments we use two cases of our approach: one preserving 46% of the information (we notate with *TD1* for keeping the most principle component) and one other preserving 80% of the information (we notate with *TD10* for the 10 most principal components).

The second step is to adjust the predictions coming from both actual and pseudo ratings. The parameter θ is responsible for this balance. To test the sensitivity of this parameter we vary θ from zero (solely rely upon pseudo ratings) to one (solely rely upon actual ratings). We test our approach using the TD1 and TD10 configurations.

As shown in Figure 4-10(a), for values of θ close to 1 we get the lowest MAE value. However, MAE is good only for the evaluation of the prediction but not for the recommendation. That is why we also use F1 to examine the impact of θ. As it is shown in Figure 4-10(b), F1 obtains the optimum values for θ between 0.85 and 0.95. As indicated in [59], content-boosted algorithms provide better results so better

quality results are derived for values of θ close to 1 but not 1. Thus in the next experiments we set θ to be 0.95.



(a) MAE (b) F1

Figure 4-10: Impact of parameter θ on TD1 and TD10

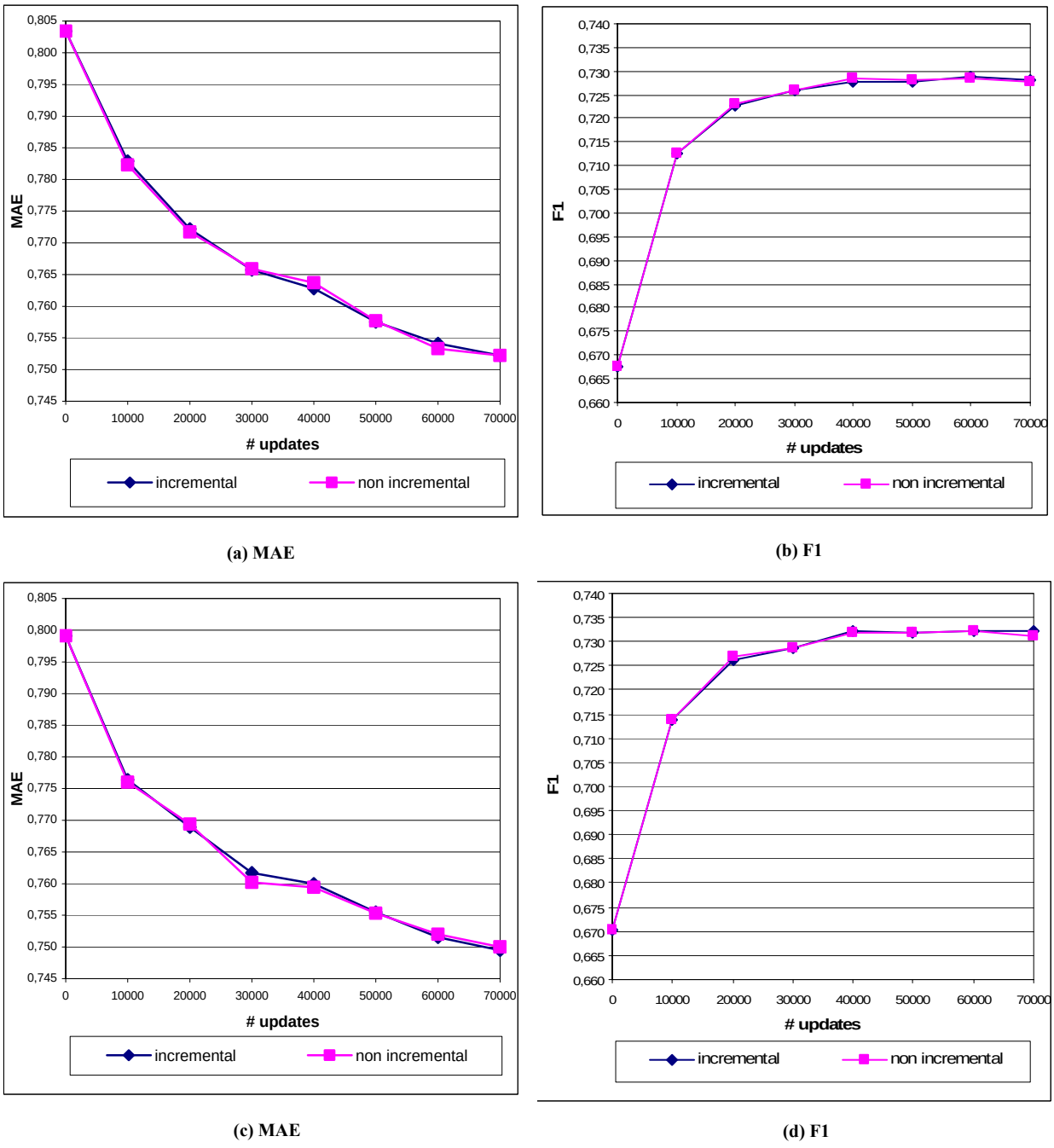
	-	10.000	20.000	30.000	40.000	50.000	60.000	70.000
ITD1	0,8034	0,7829	0,7722	0,7657	0,7628	0,7575	0,7541	0,7522
TD1		0,7823	0,7717	0,7659	0,7637	0,7577	0,7533	0,7522
ITD10	0,7991	0,7764	0,7689	0,7617	0,7600	0,7555	0,7515	0,7495
TD10		0,7760	0,7694	0,7602	0,7594	0,7553	0,7520	0,7500

(a) MAE

	-	10.000	20.000	30.000	40.000	50.000	60.000	70.000
ITD1	0,6675	0,7126	0,7228	0,7258	0,7278	0,7279	0,7287	0,7282
TD1		0,7125	0,7229	0,7258	0,7284	0,7281	0,7283	0,7278
ITD10	0,6701	0,7138	0,7263	0,7288	0,7322	0,7320	0,7322	0,7323
TD10		0,7137	0,7268	0,7285	0,7318	0,7319	0,7321	0,7312

(b) F1

Table 4-3: Comparison of incremental and non-incremental approaches while updating user-item rating matrix.



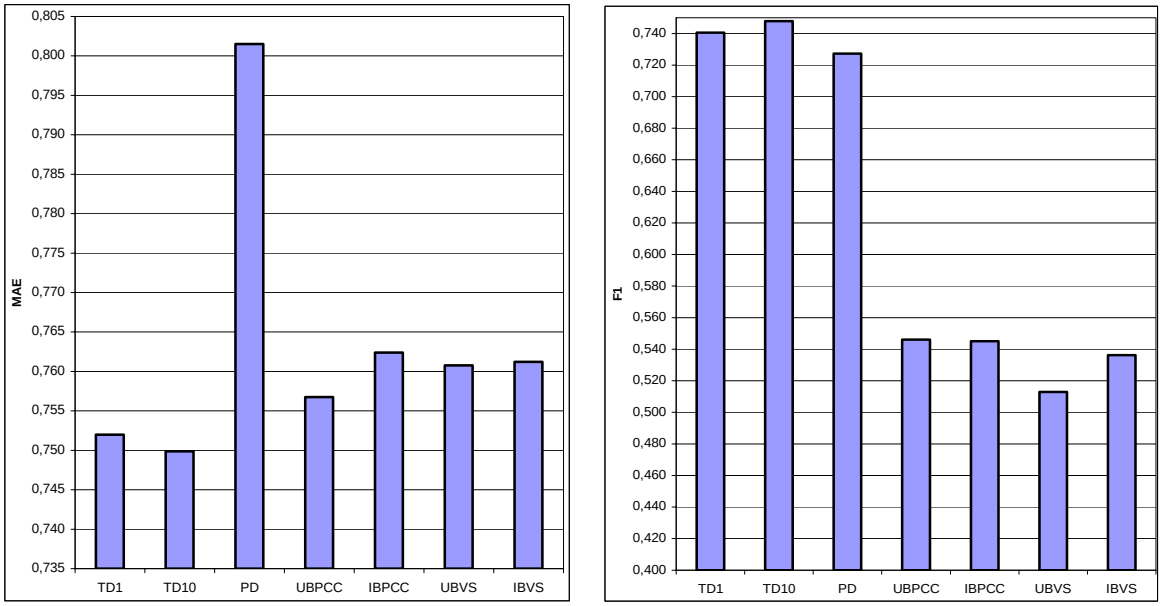
(a) MAE (b) F1

(c) MAE (d) F1

Figure 4-11: Comparison of incremental and non-incremental approaches while updating user-item rating matrix

Next, we test our technique on updating of the recommendation information. We kept an initial amount of the training set and the rest was used to update the training information. We compared TD1 and TD10 in two configurations: one which uses the incremental PCA technique to learn the new item subspace through the present one and one other which computes PCA in a batch mode (non incrementally). Results are shown in Table 4-3. Also as Figure 4-11 shows, in each diagram the two curves

almost overlap, which means that our framework provides results of the same quality with the batch-method.



(a) MAE **(b) F1**

Figure 4-12: TD1 and TD10 compared to other collaborative filtering algorithms

In experiment, we compare our TD1 and TD10 with their naïve form, PD, and also with other state-of-art memory-based algorithms, user-based and item-based PCC and VS in terms of MAE and F1. As indicated in Figure 4-12, TD1 and TD10 presented better prediction accuracy than PD regarding MAE by approximately 6%. As for F1, TD1 and TD10 provide slightly better quality results than original PD. Moreover, TD1 and TD10 outperformed PCC and VS in both their user- and item-based configurations.

4.5 Conclusions

In this chapter we have proposed a new framework for collaborative filtering, based on trends resultant of user rating activity over items. This approach makes use of an incremental PCA algorithm to track a subspace in the item space where thereafter users are mapped towards the creation of a user-trend matrix "thinner" than the initial user-rating matrix. Thus, the computations of user-to-user similarities are

carried faster than directly from user-rating matrix. But before PCA, to deal with the data missing problem we adopted a content-based approach. In pure CF a prediction cannot be made for an item, for the active user, unless it was previously rated by other users. Thus, we can further improve predictions by utilizing the content-based predictions of *other* users as well as actual ratings.

Our method can be considered twice as hybrid: First off, it consists of a hybridization due to combining a memory-based (which relies on finding the most similar ones among the past users) and a model-based approach (which develops a model about user ratings). Hybridization also comes from the fact that both actual ratings and content-based ratings are being combined towards the final prediction.

Since high dimensionality seems to be "Achilles' heel" for most of the CF-based recommendation systems, to deal with this scalability problem, we proposed an incremental background modeling technique which updates rating trends and replaces expensive vector operations with a scalar operation. Thus, we achieved to speed-up computations of high dimensional user-item matrices preserving, as shown in our experiments, high quality in recommendation formulation and, thus, negating the trade off between recommendation accuracy and response time.

Chapter 5

Conclusions

5.1 Summary

The vast volume of information flowing on the web has given rise to the need for development of more sophisticated information retrieval and filtering techniques. In e-commerce environments, recommender systems are software applications that aim at supporting the (online) user in the decision-making and buying process. The main tasks of such a system typically include the elicitation of user preferences, the construction or update of the user model, hence the generation of a personalized buying proposal, or the provision of other types of information which can be useful for the customer's decision-making process. Throughout this research, we tried to obtain a better understanding of the algorithmic foundations of recommendation technologies and to describe methodologies that could overcome certain shortcomings and expand their applicability.

At the beginning we made an introduction to recommendation algorithms formulating the recommendation problem, classifying the existing recommendation approaches and presenting specific shortcomings that need to be addressed. Some of these limitations, we dealt with, are sparsity, cold-start and scalability. Moreover we presented and discussed the work done so far towards overcoming these limitations.

Next, we proposed the use of the two basic combination schemes from theory of probabilities to overcome accuracy issues of the RS. Results have shown that a pure probabilistic combination of recommendation techniques can provide results up to 10% better than naïve linear numerical weighting of results derived from each technique. Among the probabilistic approaches, the product rule does not require no further tuning as is in the case of sum rule. However, it’s worth noticing that in most

cases the sum–rule (in its best configuration) has slightly outperformed the product–rule. The main reason is the sensitivity in errors which is intense in the latter case due to factorization of prediction techniques.

Finally, we have proposed a new framework for collaborative filtering migrating from user ratings to user trends. Information about user trends based on rating activity can be distilled by using an incremental PCA algorithm to map users into a dynamic traceable subspace of the item space. The result is the creation of a "thinner" than the original user-rating matrix for further computation of user similarities. This incremental background modeling technique updates user rating trends and replaces expensive vector operations with a scalar operation. Thus, we achieved to speed-up computations of high dimensional user-item matrices with no trade off between recommendation accuracy and response time as shown in our experiments. Regarding possible extensions, three directions arise: use of trends in item-based CF, discover of trend-based user communities and testing of the robustness of the incremental PCA algorithm in the proposed framework.

5.2 Future Extensions

Nowadays, due to increasing number of e-commerce environments on the Web, the demand for new approaches to intelligent product recommendation is higher than ever: There are more online users searching for information and purchasing items over the online channel. Actually there are more online channels, where besides the classical web, mobile devices are becoming more and more important. But, there are also more vendors, there are more products, and finally, the products and services offered on the online channels are becoming more and more complex.

Recommendation technologies can be extended in several ways that include improving the profiling of users and items, incorporating more forms of information into the recommendation process in order to provide more flexible and less intrusive types of recommendations. In the remainder of this section we describe some proposed extensions and also identify various research opportunities for developing them.

5.2.1 Towards Extending Recommendation Sources/Targets

The current generation of recommendation technologies performed well in several applications, including the ones for recommending books, CDs, and news articles [62][88]. However, these methods need to be extended for more complex types of applications, such as recommending vacations, financial services, and certain types of movie applications, in order to provide better recommendations.

As was pointed out in [7] [102] [53] [1], most of the recommendation methods produce ratings that are based on a limited user and item profiles and do not take full advantage of the information in the user's transactional histories or other available data. For example, in addition to using traditional profile features, such as keywords and simple user demographics [72] [62], more advanced profiling techniques based on *data mining rules* [30] [1], *sequences* [58], and *signatures* [23] that describe user's interests can be used to build user profiles. Similar techniques can also be used to build item profiles. Once user and item profiles are built, the most general ratings estimation function can be defined in terms of these profiles and the previously specified ratings. These enhanced profiles can be combined using the strategies we described in Chapter 3. More over, if we treat each user-item rating individually as predictor [105] we can make predictions for users on items, both with no ratings, by exploiting only demographical and content information.

On the other hand, so far, recommender systems have been applied to many fields of e-commerce in order to assist users in finding the products that best meet their preferences. However, while the application of recommender systems is well established to the search for objects, an interesting approach would be, for instance, the search of human resources for recruitment and team staffing processes.

It's worth noticing that a growing number of people make personal and professional information digitally available to others by managing profiles in CV databases, social networking platforms and other online services. Examples include platforms for the representation of private (e.g., Friendster, MySpace) and business (e.g., LinkedIn, OpenBC) networks and many of these platforms within only few months registered 7-digit numbers of users. To represent users we can create attributes, extracted from each candidate's CV, such as skills, diplomas, grades, university, etc.

5.2.2 Towards Modeling User Preferences

In Chapter 4 we used an approximation method, PCA, to dimensionality reduce the initial user-item matrix. However, there are other areas of mathematics and computer science, such as *mathematical approximation theory* [76] [65] [16], which can also contribute to developing better rating estimation methods.

The applicability of other approximation methods could be extended for estimating unknown ratings. For instance, in [20] to find the *approximate* temporal correlations of queries in real time, techniques from the theory of embeddings [48] and nearest neighbor algorithms [46] were adapted reducing so the time and space complexity of the computation. They used random hyperplanes drawn from Gaussian distribution to split item space into cells by introducing a binary mapping to a reduced dimensional space. To effectively obtain top correlated queries a clever structure using a hash-table was utilized.

One more example of an approximation-based approach constitutes *radial basis functions* [28] [87] [16]. Given a set of points and the values of an unknown function at these points, a radial basis function estimates the values of the function in the whole set of *R*. Radial basis functions have been extensively studied in the approximation theory, and their theoretical properties and utilization of radial basis functions in many practical applications have been understood very well [87] [16]. Any combination of these functions can create a radial basis function network from which we can extract features about users. Therefore, it should be interesting to extend radial basis methods for recommendation systems problems.

Bibliography

[1] ADOMAVICIUS, G. AND TUZHILIN, A.: *Expert-driven Validation of Rule-based User Models in Personalization Applications*. Data Mining and Knowledge Discovery, 5(1/2):33-58, 2001.

[2] ALI, K. AND VAN STAM, W.: TiVo: *Making Show Recommendations Using a Distributed Collaborative Filtering Architecture*. In Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM Press, Seattle, WA, USA, 394–401, 2004.

[3] AVERY, C. AND ZECKHAUSER, R.: *Recommender Systems for Evaluating Computer messages*. Communications of the ACM 40, 3 (March), 88–89, 1997.

[4] BAEZA-YATES, R. AND RIBEIRO-NETO, B.: *Modern Information Retrieval.* Addison-Wesley, 1999.

[5] BALABANOVIC, M.: *Exploring versus Exploiting when Learning User Models for Text Representation*. User Modeling and User-Adapted Interaction 8(1-2), 71-102, 1998.

[6] BALABANOVIC, M.: *An Adaptive Web Page Recommendation Service*. In Agents 97 Proceedings of the First International Conference on Autonomous Agents, Marina Del Rey, CA, pp. 378-385, 1997.

[7] BALABANOVIC, M. AND SHOHAM, Y.: *Fab: Content-based, Collaborative Recommendation.* Communication of the ACM, Mar. 1997, 40(3): 66-72.

[8] BASU, C., HIRSH, H. AND COHEN W.: *Recommendation as Classification: Using Social and Content-Based Information in Recommendation.* In Proceedings of the 15th National Conference on Artificial Intelligence, Madison, WI, pp. 714-720, 1998.

[9] BELKIN, N. AND CROFT, B.: *Information Filtering and Information Retrieval*. Communications of the ACM, 35(12):29-37, 1992.

[10] BENNETT, P. N., DUMAIS, S. T., AND HORVITZ, E.: *Probabilistic Combination of Text Classifiers Using Reliability Indicators: Models and Results*. In Proceedings of SIGIR'02. Tampere, Finland, pages 207-214, 2002.

[11] BILLSUS, D. AND PAZZANI, M.: *Learning Collaborative Information Filters.* In International Conference on Machine Learning, Morgan Kaufmann Publishers, 1998.

[12] BILLSUS, D. AND PAZZANI, M.: *User Modeling for Adaptive News Access.* User-Modeling and User-Adapted Interaction 10(2-3), 147-180, 2000.

[13] BRAND, M.: *Fast Online SVD Revisions for Lightweight Recommender Systems*. SIAM International Conference on Data Mining (SDM), May 2003.

[14] BREESE, J. S., HECKERMAN, D., AND KADIE, C.: *Empirical Analysis of Predictive Algorithms for Collaborative Filtering*. In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, 1998.

[15] BRIN, S. AND PAGE, L.: *The Anatomy of a Large-Scale Hypertextual {Web} Search Engine*. Computer Network and ISDN Systems, 30(1-7), 107-117, 1998.

[16] BUHMANN, M. D.: *Approximation and Interpolation with Radial Functions. In Multivariate Approximation and Applications*. Eds. N. Dyn, D. Leviatan, D. Levin, and A. Pinkus. Cambridge University Press, 2001.

[17] BURKE, R.: *Knowledge-based Recommender Systems.* In A. Kent (ed.), Encyclopedia of Library and Information Systems. Vol. 69, Supplement 32. New York: Marcel Dekker, 2000.

[18] BURKE, R.: *Hybrid Recommender Systems: Survey and Experiment*. User Modeling and User-Adapted Interaction 12(4): 331-370; Nov 2002.

[19] BURKE, R., HAMMOND, K., AND YOUNG, B.: *The FindMe Approach to Assisted Browsing*. IEEE Expert, 12 (4), 32-40, 1997.

[20] CHIEN, S., AND IMMORLICA, N.: *Semantic Similarity Between Search Engine Queries Using Temporal Correlation*. Proceedings of the 14th International Conference on WWW, Chiba, Japan, p 2 – 11, 2005.

[21] CLAYPOOL, M., GOKHALE, A., MIRANDA, T., MURNIKOV, P., NETES, D. AND SARTIN, M.: *Combining Content-Based and Collaborative Filters in an Online Newspaper.* In SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation. Berkeley, CA, 1999.

[22] CONDLIFF, M. K., LEWIS, D. D., MADIGAN, D. AND POSSE, C.: *Bayesian Mixed-Effects Models for Recommender Systems.* In SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation. Berkeley, CA, 1999.

[23] CORTES, C., FISHER, K., PREGIBON, D., ROGERS, A., AND SMITH, HANCOCK, F.: *A Language for Extracting Signatures from Data Streams*. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000.

[24] COSLEY, D., LAWRENCE, S., PENNOCK, D. M. REFEREE: *An Open Framework for Practical Testing of Recommender Systems using Research Index.* Proc. of the 28th Very Large Data Bases Conference, 2002.

[25] DEERWESTER, S., DUMAIS, S. T., FURNAS, G. W., LANDAUER, T. K., HARSHMAN, R.: *Indexing by Latent Semantic Analysis.* Journal of the American Society for Information Science 41(6), 1990.

[26] DELGADO, J. AND ISHII, N.: *Memory-based Weighted-Majority Prediction for Recommender Systems.* In ACM SIGIR'99 Workshop on Recommender Systems: Algorithms and Evaluation, 1999.

[27] DESHPANDE, M. AND KARYPIS, G.: *Item-based Top-N Recommendation Algorithms.* ACM Transactions on Information Systems 22, 1, 143–177, 2004.

[28] DUCHON, J.: *Splines Minimizing Rotation-Invariate Semi-Norms in Sobolev Spaces*. In Constructive Theory of Functions of Several Variables, ed. W. Schempp and Zeller, pp. 85-100, Springer 1979.

[29] DUDA, R. O., HART, P. E. AND STORK, D. G.: *Pattern Classification*, John Wiley and Sons, Inc., 2001.

[30] FAWSETT, T., AND PROVOST, F.: *Combining Data Mining and Machine Learning for Efficient User Profiling*. In Proceedings of the Second Intenational Conference On Knowledge Discovery and Data Mining (KDD-96), 1996.

[31] GAUL, W. AND SCHMIDT-THIEME, L.: *Recommender Systems Based on User Navigational Behaviour in the Internet*. Behaviormetrika 29, 1, pages 1–22, 2002.

[32] GLANCE, N., ARREGUI, D. AND DARDENNE, M.: *Making Recommender Systems Work for Organizations*. In Proceedings of PAAM'99, London, UK, April 19-21, 1999.

[33] GOLBECK J.: *Personalizing Applications through Integration of Inferred Trust Values in Semantic Web-based Social Networks.* Semantic Network Analysis Workshop at the 4th International Semantic Web Conference, 2005.

[34] GOLDBERG, D., NICHOLS, D., OKI, B. M., AND TERRY, D.: *Using Collaborative Filtering to Weave an Information Tapestry*. Communications of the ACM, 35(12):61-70, 1992.

[35] GOLDBERG, K., ROEDER, T., GUPTA, D., AND PERKINS, C.: *Eigentaste: A Constant Time Collaborative Filtering Algorithm*. Information Retrieval Journal, 4(2):133-151, July 2001.

[36] GOOD, N., SCHAFER, B., KONSTAN, J., BORCHERS, A., SARWAR, B., HERLOCKER, J., AND RIEDL, J.: *Combining collaborative filtering with personal agents for better recommendations.* In Proceedings of the 16th National Conference on Artificial Intelligence and Innovative Applications of Artificial Intelligence. American Association for Artificial Intelligence, Orlando, FL, USA, 439–446, 1999.

[37] GUTTMAN, ROBERT H.: *Merchant Differentiation through Integrative Negotiation in Agent-mediated Electronic Commerce.* Master's Thesis, School of Architecture

and Planning, Program in Media Arts and Sciences, Massachusetts Institute of Technology, 1998.

[38] HERLOCKER, J. L., KONSTAN, J. A., BORCHERS, A., AND RIEDL, J.: *An Algorithmic Framework for Performing Collaborative Filtering.* In Proceeding of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1999.

[39] HERLOCKER, J.L., KONSTAN, J. A. AND RIEDL, J.: *Explaining Collaborative Filtering Recommendations*. In Proceedings of CSCW'2000. ACM, Philadelphia, pages 241-250, 2000.

[40] HERLOCKER, J.L., KONSTAN, J. A., TERVEEN, L. AND RIEDL, J.: *Evaluating Collaborative Filtering Recommender Systems*. ACM Transactions on Information Systems (TOIS), 22 (1), 2004.

[41] HIEMSTRA, D.: *Term-specific Smoothing for the Language Modeling Approach to Information Retrieval: the Importance of a Query Term.* In Proceedings of SIGIR, p. 35-41, 2002.

[42] HILL, W., STEAD, L., ROSENSTEIN, M. AND FURNAS, G.: *Recommending and Evaluating Choices in a Virtual Community of Use*. In Proceedings of Computer Human Interaction, 1995.

[43] HOFMANN, T.: *Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis.* In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2003.

[44] HOUSEMAN, E. M. AND KASKELA, D. E.: *State of the Art of Selective Dissemination of Information*. IEEE Trans Eng Writing Speech III, pages 78-83, 1970.

[45] HUANG, Z., CHEN, H., ZENG, D.: *Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering*. ACM Transactions on Information Systems, 22 (1), 2004.

[46] INDYK, P., AND MOTWANI, R.: *Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality.* In Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, 1998.

[47] JACKSON, P.: *Introduction to Expert Systems*. Addison-Wesley, Reading, MA., 1990.

[48] JOHNSON, W., AND LINDENSTRAUSS, J.: *Extensions of Lipschitz Maps into a Hilbert Space.* Contemporary Mathematics, 26:189–206, 1984.

[49] JUNG, S. Y., AND KIM, T.: *An Incremental Similarity Computation Method in Agglomerative Hierarchical Clustering*. In Proceedings of the International Symposium on Advanced Intelligent Systems, 2001.

[50] KARYPIS, G.: *Evaluation of item-based top-N recommendation algorithms.* In Proceedings of the Tenth ACM CIKM International Conference on Information and Knowledge Management. ACM Press, Atlanta, GA, USA, 247–254, 2001.

[51] KITTLER, J., HATEF, M., DUIN, R. P. M., AND MATAS, J.: *On Combining Classifiers*. IEEE Transaction Pattern Analysis Machine Intelligence, 20(3):226-239, 1998.

[52] KLEINBERG, J., PAPADIMITRIOU, C. H., RAGHAVAN, P.: *On the Value of Private Information.* In Proceedings of 8th Conference on Theoretical Aspects of Reasoning about Knowledge, 2001.

[53] KONSTAN, J. A., MILLER, B. N., MALTZ, D., HERLOCKER, J. L., GORDON, L. R., AND RIEDL, J.: *GroupLens: Applying collaborative filtering to Usenet news*. Communications of the ACM, 40(3):77-87, 1997.

[54] KRULWICH, B.: *Lifestyle Finder: Intelligent User Profiling Using Large-Scale Demographic Data*. Artificial Intelligence Magazine 18 (2), 37-45, 1997.

[55] LI., Y.: *On Incremental and Robust Subspace Learning.* Pattern recognition, 37:1509–1518, 2004.

[56] LINDEN, G., SMITH, B., AND YORK, J.: *Amazon.com recommendations: Item-to-item collaborative filtering*. IEEE Internet Computing 4, 1 (January), 2003.

[57] LITTLESTONE, N. AND WARMUTH, M.: *The Weighted Majority Algorithm*. Information and Computation 108 (2), 212-261, 1994.

[58] MANNILA, H., TOIVONEN, H., AND VERKAMO, A. I.: *Discovering Frequent Episodes in Sequences*. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95), 1995.

[59] MELVILLE, P., MOONEY, R. J., AND NAGARAJAN, R.: *Content-Boosted Collaborative Filtering for Improved Recommendations*. In Proceedings of the Eighteenth National Conference on Artificial Intelligence, 2002.

[60] MIDDLETON, S., SHADBOLT, N., AND DE ROURE, D.: *Ontological User Profiling in Recommender Systems*. ACM Transactions on Information Systems 22, 1, pages 54–88, 2004.

[61] MOONEY, R. J., BENNETT, P. N., AND ROY, L.: *Book Recommending Using Text Categorization with Extracted Information.* In Recommender Systems. Papers from 1998 Workshop. Technical Report WS-98-08. AAAI Press, 1998.

[62] MOONEY, R. J. AND ROY, L.: *Content-Based Book Recommending Using Learning for Text Categorization*. In SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation. Berkeley, CA, 1999.

[63] NAKAMURA, A. AND ABE, N.: *Collaborative Filtering Using Weighted Majority Prediction Algorithms*. In Proceedings of the 15th International Conf. Machine Learning, 1998.

[64] NICHOLS, D.: *Implicit Rating and Filtering*. In Proceedings of the 5th DELOS Workshop on Filtering and Collaborative Filtering. ERCIM, Budapest, Hungary, pages 31– 36, 1998.

[65] NURNBERGER, G.: *Approximation by Spline Functions*. Springer-Verlag, 1989.

[66] OARD. D.W.: *The State of the Art in Text Filtering*. User Modeling and User Adapted Interaction, 7(3)141-178, 1997.

[67] PAPAGELIS, M. AND PLEXOUSAKIS, D.: *Qualitative Analysis of User-based and Item-based Prediction Algorithms for Recommendation Agents.* Eighth International Workshop on Cooperative Information Agents, 2004.

[68] PAPAGELIS, M., PLEXOUSAKIS, KUTSURAS, T.: *A method for alleviating the Sparsity Problem in Collaborative Filtering Using Trust Inferences*. Proceedings of the 3rd International Conference on Trust Management, 2005.

[69] PAPAGELIS, M., PLEXOUSAKIS, D., ROUSIDIS, I., AND THEOHAROPOULOS, E.: *Qualitative Analysis of User-based and Item-based Prediction Algorithms for Recommendation Systems*. 3rd Hellenic Data Management Symposium, 2004.

[70] PAPAGELIS, M., ROUSIDIS, I., PLEXOUSAKIS, D., AND THEOHAROPOULOS, E.: *Incremental Collaborative Filtering for Highly-Scalable Recommendation algorithms.* 15th International Symposium on Methodologies of Intelligent Systems, 2005.

[71] PAZZANI, M. J.: *A Framework for Collaborative, Content-Based and Demographic Filtering*. Artificial Intelligence Review, 13 (5/6), 393-408, 1999.

[72] PAZZANI, M. J. AND BILLSUS, D.: *Learning and Revising User Profiles: The identification of interesting web sites*. Machine Learning, 27:313-331, 1997.

[73] PENNOCK, D. M. AND HORVITZ, E.: *Collaborative Filtering by Personality Diagnosis: A Hybrid Memory- and Model-based Approach*. In Proceedings of UAI, 2000.

[74] PITSILIS, G., AND MARSHALL, L. F.: *Trust as a Key to Improving Recommendation Systems*. In Proceedings of 3rd International Conference iTrust 2005, Paris France, May 2005.

[75] POPESCUL, A., UNGAR, L. H., PENNOCK, D.M., LAWRENCE, S.: *Probabilistic Models for Unified Collaborative and Content-Based Recommendation in*

Sparse-Data Environments. In Proceedings of Uncertainty in Artificial Intelligence, 2001.

[76] POWELL, M. J. D. *Approximation Theory and Methods,* Cambridge University Press, 1981

[77] RASHID, A. M., ALBERT, I., COSLEY, D., LAM, S. K., MCNEE, S. M., KONSTAN, J. A., AND RIEDL, J.: *Getting to Know You: Learning New User Preferences in Recommender Systems.* In Proceedings of the International Conference on Intelligent User Interfaces, 2002.

[78] RASHID, A. M., LAM, S. K., KARYPIS G. AND RIEDL J.: ClustKNN: *A Highly Scalable Hybrid Model- & Memory-Based CF Algorithm.* In Proceedings of *WEBKDD 2006*, August 20, 2006, Philadelphia, Pennsylvania, USA, 2006.

[79] RESNICK, P., IAKOVOU, N., SUSHAK, M., BERGSTROM, P., AND RIEDL, J. GroupLens: *An Open Architecture for Collaborative Filtering of Netnews*. In Proceedings of the Computer Supported Cooperative Work Conference, 1994.

[80] RESNICK, P. AND VARIAN, H. R.: *Recommender Systems.* Communications of the ACM, 40 (3), 56-58, 1997.

[81] RICH, E.: *User Modeling via Stereotypes*. Cognitive Science 3, 329-354, 1979.

[82] ROCCHIO, JR., J.: *Relevance Feedback in Information Retrieval*. In The SMART System – Experiments in Automatic Document Processing, New York: Prentice Hall, pp. 313-323, 1971.

[83] SALTON , G.: *Automatic Text Processing.* Addison-Wesley, 1989.

[84] SARWAR, B., KARYPIS, G., KONSTAN, J., AND RIEDL, J.: *Item-based Collaborative Filtering Recommendation Algorithms*. In Proceedings of the 10th International WWW Conference, 2001.

[85] SARWAR, B., KARYPIS, G., KONSTAN, J., AND RIEDL, J.: *Application of Dimensionality Reduction in Recommender Systems.* In ACM WebKDD Workshop. Boston, MA, USA, 2000.

[86] SARWAR, B. M., KONSTAN, J. A., BORCHERS, A., HERLOCKER, J. MILLER, B. AND RIEDL, J.: *Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System*. In Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work, Seattle, WA, pp. 345-354, 1998.

[87] SCHABACK, R. AND WENDLAND, H.: *Characterization and construction of radial basis functions*. In Multivariate Approximation and Applications. Eds. N. Dyn, D. Leviatan, D. Levin and A. Pinkus. Cambridge University Press, 2001.

[88] SCHAFER, J. B., KONSTAN, J. AND RIEDL, J.: *Recommender Systems in E-Commerce*. In Proceedings of the First ACM Conference on Electronic Commerce, Denver, CO, pp. 158-166, 1999.

[89] SCHEIN, A. I., POPESCUL, A., UNGAR, L. H., AND PENNOCK, D. M.: *Methods and metrics for cold-start recommendations.* In Proceedings of the 25th Annual International ACM SIGIR Conference, 2002.

[90] SCHMITT, S. AND BERGMANN, R.: *Applying Case-based Reasoning Technology for Product Selection and Customization in Electronic Commerce Environments.* 12th Bled Electronic Commerce Conference. Bled, Slovenia, June 7-9, 1999.

[91] SCHWAB, I., KOBSA, A. AND KOYCHEV, I.: *Learning User Interests through Positive Examples Using Content Analysis and Collaborative Filtering*. User Modeling and User-Adapted Interaction. draft from Fraunhofer Institute for Applied Information Technology, Germany, 2001.

[92] SHARDANAND, U. AND MAES, P.: *Social Information Filtering: Algorithms for Automated "Word of Mouth"*. In Proceedings of Human factors in computing systems. ACM, New York, pages 210-217, 1995.

[93] SHETH, B. AND MAES P.: *Evolving Agents for Personalized Information Filtering.* In Proceedings of the 9th IEEE Conference on Artificial Intelligence for Applications, 1993.

[94] SMYTH, B. AND COTTER, P.: *A Personalized TV Listings Service for the Digital TV Age*. Knowledge-Based Systems 13: 53-59, 2000.

[95] SREBRO, N., AND JAAKKOLA, T.: *Weighted Low Rank Approximation*, 2003.

[96] SYMEONIDIS, P., NANOPOULOS, A., PAPADOPOULOS, A., AND MANOLOPOULOS, Y.: *Collaborative Filtering based on User Trends*. Proceedings of the 30th Annual Conference of the German Classification Society (GfKl 2006), March 8-10, Berlin, Germany, 2006A.

[97] SYMEONIDIS, P., NANOPOULOS, A., PAPADOPOULOS, A., AND MANOLOPOULOS, Y.: *Scalable Collaborative Filtering Based on Latent Semantic Indexing*. Proceedings of the AAAI Workshop on Intelligent Techniques for Web Personalization (ITWP 2006), July 16-20 , Boston , United States 2006B.

[98] TERVEEN, L., HILL, W., AMENTO, B., MCDONALD, D., AND CRETER, J.: PHOAKS: *A System for Sharing Recommendations*. Communications of the ACM, 40(3):59-62, 1997.

[99] TOWLE, B. AND QUINN, C.: *Knowledge Based Recommender Systems Using Explicit User Models.* In Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, AAAI Technical Report WS-00-04. pp. 74-77. Menlo Park, CA: AAAI Press, 2000.

[100] TOYAMA, K. AND HORVITZ, E.: *Bayesian Modality Fusion: Probabilistic Integration of Multiple Vision Algorithms for Head Tracking*. In Proceedings of ACCV, 4th Asian Conference on Computer Vision, 2000.

[101] TRAN, T. AND COHEN, R.: *Hybrid Recommender Systems for Electronic Commerce.* In Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, AAAI Technical Report WS-00-04. pp. 78-83. Menlo Park, CA: AAAI Press, 2000.

[102] UNGAR, L. H., AND FOSTER, D. P.: *Clustering Methods For Collaborative Filtering In Recommender Systems*. Papers from 1998 Workshop. Technical Report WS-98-08. AAAI Press, 1998.

[103] VAN SETTEN, M.: *Experiments with a Recommendation Technique that Learns Category Interests*. In Proceedings of IADIS WWW/Internet 2002, Lisbon, Portugal, pages 722-725, 2002.

[104] WANG, J., DE VRIES, A. P., AND REINDERS, M.J.: *A User-Item Relevance Model for log-based Collaborative Filtering*. In Proceedings of ECIR06, London, UK, 2006A.

[105] WANG, J., DE VRIES, A. P., AND REINDERS, M.J.: *Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion*. In Proceedings of SIGIR, 2006B.

[106] WASFI, A. M.: *Collecting User Access Patterns for Building User Profiles and Collaborative Filtering*. In Proceedings of the 1999 International Conference on Intelligent User Interfaces, Redondo Beach, CA, pp. 57-64, 1999.

[107] XUE, G.-R., LIN, C., YANG, Q., XI, W., ZENG, H.-J., YU, Y. AND CHEN, Z.: *Scalable Collaborative Filtering using Cluster-based Smoothing.* In Proceedings of SIGIR, 2005.

[108] YU, K., XU, X., TAO, J., ESTER, M., AND KRIEGEL, H.-P.: *Instance Selection Techniques for Memory- Based Collaborative Filtering.* In Proceedings of Second SIAM International Conference on Data Mining, 2002.

[109] ZENG, C., XING, C., AND ZHOU, L.: *Similarity Measure and Instance Selection for Collaborative Filtering*. In Proceedings of WWW, 2003.

[110] GroupLens Research Project: http://www.grouplens.org/

[111] Matlab – The Language of Technical Computing: http://www.mathworks.com/products/matlab/